Journal of Engineering Science and Technology
Vol. 16, No. 3 (2021) 1991 - 2000

ELMAN RECURRENT NEURAL NETWORK FOR ASPECT BASED SENTIMENT ANALYSIS

NELLY INDRIANI WIDIASTUTI*, MAULVI INAYAT ALI

Informatics Engineering Department, Universitas Komputer Indonesia,
Jalan Dipati Ukur No.112-116 Bandung 40132, Jawa Barat, Indonesia
*Corresponding Author: nelly.indriani@email.unikom.ac.id

Abstract

Aspect based sentiment analysis (ABSA) is one of the domains of opinion mining cases which aims to detect the polarity of written text based on certain aspects. The purpose of this study was to determine the accuracy value of Elman RNN in the case of ABSA. The method used is divided into two main processes, namely pre-process and sentiment detection. Before conducting the training, the input data in the form of restaurant reviews in Indonesian went through the pre-processing process. In the data review, case folding, filtering, word normalization, tokenization, stop word removal, the addition of token aspects (price, taste, atmosphere) was carried out, building a word dictionary, and forming one-hot encoding. In the polarity detection process, training and testing use the ERNN algorithm. The data used were 1584 sentences of Indonesian restaurant reviews and were tested on 422 data. Based on the test results, Elman RNN got the best accuracy of 81.22% and an f1 score of 82.78%. For the social analytic system developer, these results show evidence that the ERNN is promising to be used in detecting the polarity of a restaurant review.

Keywords: Aspect based sentiment analysis, Indonesian language, Pre-processing, Recurrent neural network, Semantic evaluation.

1. Introduction

Aspect based sentiment analysis (ABSA) is one of the opinion mining case domains to detect the polarity of written texts based on certain aspects. This field analyses opinions, sentiments, evaluations, assessments of attitudes and emotions towards entities such as products, services, organizations, individuals, events, topics, and other attributes. Sentiment analysis and opinion mining focus on opinions that express positive and negative sentiments [1]. Classification of opinion texts at the document or sentence level is often not enough because that level can only identify general sentiment. If it is assumed a document or sentence has a positive sentiment, it does not mean all aspects in the document or sentence are all positive as well as for documents or sentences that contain negative sentiment. For a more complete analysis, it is necessary to find aspects and determine positive or negative sentiments in each aspect [1]. Various algorithms have been used to analyse sentiment, one of which is RNN. RNN is one of the Deep Learning algorithms that has a network with loops, which allows information to survive in the network. RNN has a feedback connection to the network itself that allows activation to flow back in a loop to learn sequences and information to survive [2].

Sentiment analysis based on aspects has recently become an interesting topic for researchers, especially in the field of sentiment analysis. SemEval is a workshop in the field of natural language processing, one of which is ABSA at SemEval 2014, SemEval 2015, and SemEval 2016. Several studies in the domain of sentiment analysis using deep learning algorithms have been carried out including Tran [3] using BiGRU RNN with word embedding, SenticNet, POS Tag, and Distance features which turned out to only produce accuracy of 78.7%. Another RNN architecture used by Tarasov [4] with the word embedding feature which only produces accuracy for Bidirectional RNN (65.10%) and LSTM (69.70%). From some researches on ABSA, using CNN and some RNN architectures. The resulting accuracy is still below 80%, which can still be improved again.

In our previous research, we reviewed several studies using deep learning in the domain of text mining [5]. In the paper explains that one that affects the performance of neural networks is network architecture, in addition to that other research in the case of fine-grained mining opinion Liu et al. [6] compared some RNN architectures with the SENNA embedding feature which resulted in an accuracy of 81.36% for Elman RNN, 79.43% for LSTM and 78.83% for Jordan RNN. The Elman architecture outperforms other RNN architectures without doing too much feature engineering. RNN is also very good for the need to store feedback information [2].

Based on the description that has been presented, this study aims to identify positive, negative, or non-sentiment towards a restaurant by using the aspects that appear in every review in Indonesian.

In this study using Elman RNN to classify sentiment in training and detection. The training data uses the input feature in the form of a word token plus an aspect token that appears in a review.

2. Research Method

The study was conducted using data from Cahyadi and Khodra [7] research in using Bidirectional LSTM and CNN. We divided the data into 1584 sentences for training data and 422 sentences for test data. The data is a review of restaurants whose

aspects have been determined namely, Food (Fi), Price (Pi), Service (Si), and Ambience (Ai). All the review data is preprocessed and then aspect tokens are added, word dictionary development and one hot encoding are done. From the pre-processing results, one hot encoding is the input for ERNN.

2.1. Aspect based sentiment analysis

Sentiment analysis is part of mining text that focuses on extracting a comment on something in the form of goods, services, someone, or an event. Sentiment analysis consists of 3 levels, namely document level, sentence level, entity, and aspect level. For a more complete analysis, it is necessary to find aspects and determine positive or negative sentiments in each aspect [1].

There are two main tasks in sentiment analysis based on aspects, namely aspect extraction, and aspect classification. In the aspect extraction, a sentiment is determined by the kind of aspect being discussed. Whereas in the aspect of sentiment classification, a sentiment is classified as positive, negative, or neutral for that aspect.

2.2. Process model

In the implementation of ERNN in the case of ABSA, aspects of the training data have been determined together with the sentiment in each sentence. Before entering the ERNN training for sentiment classification, pre-processing needs to be done. This preprocessing stage includes the case of folding, filtering, word normalization, tokenization, stop word removal, adding token aspects, building a word dictionary, and one hot encoding. In addition to the sentence in the training data, the sentence in the test data is also preprocessed before being tested. The process model to be built can be seen in Fig. 1 as follow.

All training and testing data will go through the pre-processing stage. The difference for testing data is that there is no word dictionary development process because it was already made at the preprocessing for the training stage. ERNN test results in the form of sentiment classification based on the aspects and accuracy obtained.

2.3. Preprocessing

Preprocessing is required to process non-numeric data. This has an effect even though in this study there is not much language processing [8]. The preprocessing phase consists of case folding, filtering, word normalization, tokenization, stop word removal, adding of aspect tokens, word dictionary development, and one hot encoding.

- Case folding is the process of converting the entire text into uppercase or lowercase. In this study, all letters in the input data are converted into lowercase letters to facilitate word recognition in the main algorithm.
- Filtering is the process of removing symbols that are not needed in sentences such as punctuation, numbers, and emoticons. This process is done to eliminate the noise contained in the sentence so as not to disturb the main process.
- Word Normalization is the process of changing nonstandard words into standard words. In this process, non-standard words like the word 'enakkkk' will be changed to 'enak' and the words 'ga', 'engga', 'gak', 'enggak' and 'gk' to 'tidak' so that in the process of forming a dictionary, the word is not considered different. In this study, word normalization uses the inaNLP library [9].

- Tokenization is the process of separating words into sentences to be sentence tokens. In this study, the separation is done by relying on the space character in the sentence [10].
- Stop word Removal of word removal contained in the stop word list. The word "stop word" usually refers to the most common words like the words ‘dan’, ‘dengan’, ‘dalam’, and others. In this study, the stop word removal process used the inaNLP library [9].
- The addition of aspect tokens is the process of adding tokens that represent aspects of each sentence. In this case, the aspects used to consist of aspects of FOOD, PRICE, SERVICE, AND AMBIENCE. This aspect is attached to each sentence so that it will have 4 aspect tokens at the end of the sentence. The addition of this aspect token is intended so that the ERNN algorithm (many to many) can predict the aspects in the sentence. Information for each aspect token can be seen in Table 1.

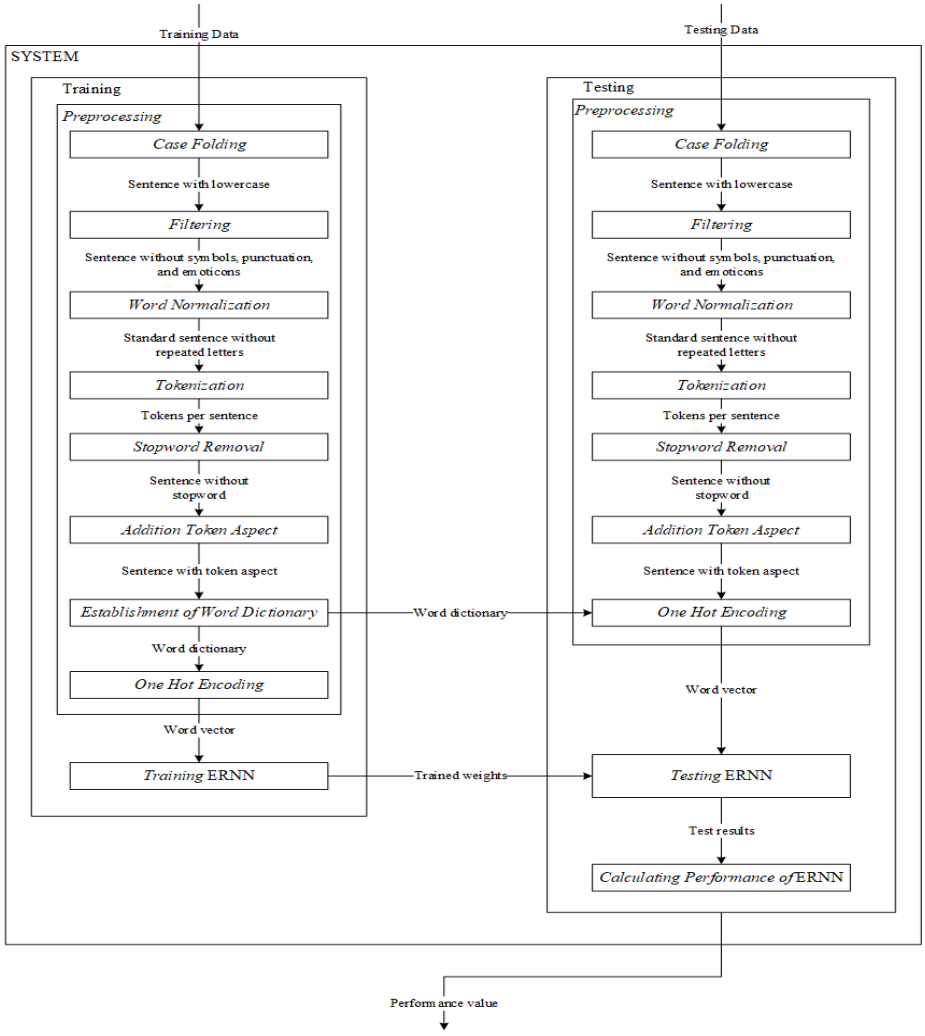

Fig. 1. Process model.

Table 1. Aspect token.

Aspect token	Definition
<FOOD>	To predict 'food' sentiment from a restaurant in a sentence.
<PRICE>	To predict 'price' sentiment from a restaurant in a sentence.
<SERVICE>	To predict 'service' sentiment from a restaurant in a sentence.
<AMBIENCE>	To predict the sentiment of the place / atmosphere of the restaurant in the sentence.

- Word dictionary development is the process of taking all the different tokens from training data to form a unique word dictionary. This dictionary was formed because to do the vectorization of tokens in the one-hot encoding process relying on the word dictionary. In the construction of the word dictionary, there will be the addition of the <UNKNOWN> token where the token works to overcome unseen words in the test data [11]. Unseen words are words that never existed in the training process) so that if there are words in the test data that are not in the training data word dictionary then it will be considered a <UNKNOWN> token. Large K words size dictionary will be equal to the number of unique or different words in the training data.
- The machine learning algorithm cannot process text data directly. Therefore, it is necessary to convert text data into numerical data so that it can be processed by the ERNN algorithm. One hot encoding will give ID to each token. ID is a representation of a number that will be a vector. The number of ID is the position of the word in the built-in dictionary. After passing the one-hot encoding stage, each token will be a K-sized vector where all elements are 0 except for only one element that is 1, that is, the position of the same element as the word in the word dictionary.

2.4. Elman recurrent neural network

ERNN is one of the deep neural networks that can process information in stages, on a normal neural network all inputs and outputs do not depend on each other, but ERNN has a memory that stores previous information and can use previously recorded information [2]. The sequence length of each RNN varies. In general, the RNN architecture can be seen in Fig. 2.

Figure 2 shows ERNN in unrolled to the full network (full network). By unrolling the RNN, we use the entire network on complete order. For example, if the sequence has 1 sentence with 5 words, the network will be opened in a 5-layer neural network which is one layer for each word.

The training stages in ERNN can be seen in Fig 3. The process consists of the initial weight initialization stages, take some random data, forward propagation ERNN, backward propagation ERNN, and calculate new weights.

2.4.1. Initialization weight

Initial weights are usually initializing randomly. As a result, when ERNN performs an error calculation at the initial weight, there will be a large error. This will make ERNN far from actual predictions. To overcome the problem, ERNN conducts the training to renewing the initial weight on each epoch. The training is expecting to make errors smaller. In ERNN, 3 weights will be used in the calculation, namely, *U*,

W, and *V*. The initialization techniques used in this study are as follows: [12]: *U* are random numbers that are at intervals $-\frac{1}{\sqrt{I}}$ dan $\frac{1}{\sqrt{I}}$. W are random numbers that are at intervals $-\frac{1}{\sqrt{H}}$ dan $\frac{1}{\sqrt{H}}$. V are random numbers that are at intervals $-\frac{1}{\sqrt{H}}$ dan $\frac{1}{\sqrt{H}}$.

2.4.2. Minibatch stochastic gradient descent

In this study, Minibatch Stochastic Gradient Descent is using in obtaining the calculation of the derivative. The process only takes part of the data in calculating the value of the child. The SGD mini batch is used so that training is more efficient in terms of time and computing than using all the training data in finding derivative values. Retrieval of some data is done every epoch, in this study the mini batch sizes used were 256, 512, and 1024.

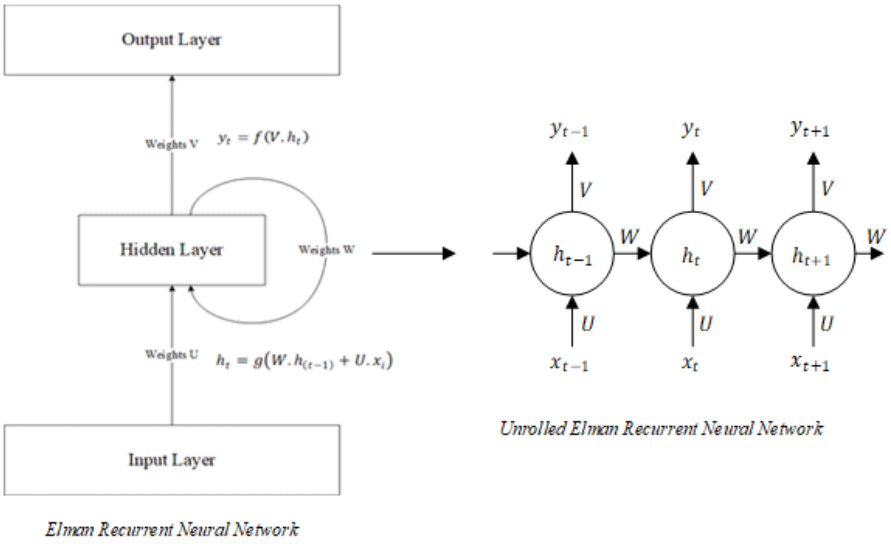

Fig. 2. Architecture ERNN.

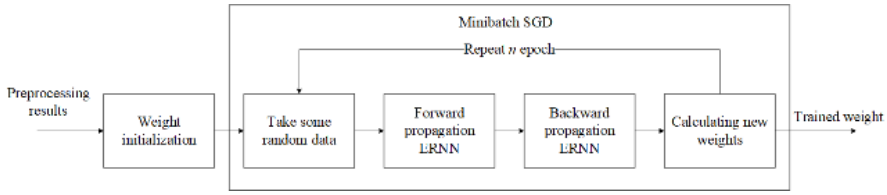

Fig. 3. ERNN training.

2.4.3. Forward propagation ERNN

Between the ERNN training processes, the forward propagation process is the initial process of the ERNN algorithm where each token of the sentence will enter one by one into the ERNN algorithm. Each token will enter from the input layer to the hidden layer from the last to the output layer. At the output layer, there will be an error calculation using cross-entropy. In forward propagation, ERNN uses the formula input layer to hidden layer as in Eq. (1).

$$h_t = g(W.h_{(t-1)} + U.x_i) \quad (1)$$

The activation function used in this study in the hidden layer is tanh, while the *t*-time output layer value is obtained from Eq. (2).

$$y_t = f(V.h_t) \quad (2)$$

2.4.4. Backward propagation ERNN

The result of forward propagation and the error is still large, we will update the weight by doing backward propagation so that the error value becomes smaller. Weights that will be updated are *U*, *W*, and *V* where to find the derivative value of three pieces, i.e., $\frac{\partial Error}{U}$, $\frac{\partial Error}{W}$, dan $\frac{\partial Error}{V}$. Backward propagation in this research is Backpropagation Through Time (BPTT). This method works by unrolling (unfolding) each input layer then the error value is accumulated for each timestep *t* [13]. In ERNN, BPTT will calculate the weights *U*, *W*, and *V*. The reduction weight formula *V* can be seen in Eq. (3 and 4).

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \quad (3)$$

$$\frac{\partial E_t}{\partial V} = (y_t - label_t) \otimes h_t \quad (4)$$

In calculating the derivative weighted of *W*, each hidden state h_t is connected to the previously hidden state $h_{(t-1)}$. It happens until the first hidden state h_1 connected to h_0. In decreasing the weight of *W*, it is necessary to pay attention to all hidden states as in Eq. (5, 6 and 7).

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \quad (5)$$

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-1)}}\frac{\partial h_{(t-1)}}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-2)}}\frac{\partial h_{(t-2)}}{\partial W}\right) + \cdots + \left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial W}\right) \quad (6)$$

$$\frac{\partial h_t}{\partial W} = \delta_t \otimes h_{t-1} \quad (7)$$

and to calculate δ_t, we can see in Eq. (8).

$$\delta_t = \left[V \ .(y_t - label_t)\right] \times \left(1 - {h_t}^2\right) \quad (8)$$

On the decrease in weight *U* has similarities with a decrease in weight, *W*. The difference is if the weight of *W* decreases to h_t while the weight of *U* decreases with x_t as in Eq. (9, 10 and 11).

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \quad (9)$$

$$\frac{\partial E_t}{\partial U} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U}\right) + \left(\frac{\partial E_t}{\partial h_{(t-1)}}\frac{\partial h_{(t-1)}}{\partial U}\right) + \left(\frac{\partial E_t}{\partial h_{(t-2)}}\frac{\partial h_{(t-2)}}{\partial U}\right) + \cdots + \left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial U}\right) \quad (10)$$

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U} = \delta_t \otimes x_t \quad (11)$$

3. Results and Discussion

In this study, parameter testing conducted in 1584 sentences of training data and 422 sentences of test data. Tests carried out with the composition of the data as shown in Table 2.

Table 2. Data composition.

Aspect	Non-Sentiment	Positive	Negative
Food	1030	493	61
Price	1209	232	143
Service	1291	239	54
Ambience	1120	401	63

At the stage of performance testing using the accuracy method and F1 score. The best parameter values obtained from performance testing. some references are suggesting their best parameters. all parameters are in Table 3 as follows.

Table 3. Testing parameters.

Parameters	Values	
	Tested	The best
Learning rate [14]	0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001	0.05
Total hidden state [15]	12, 22, 32, 42, 52	32
Minibatch Size [16]	256, 512, 1024	1024
Epoch [17]	100	100
BPTT truncate	2, 4, 6, 8, 10	10

Most hidden layers, activation or training functions that affect the overall performance of the neural network algorithm [5]. This also happened in tested cases. From each parameter, the best value is selecting by using the basic parameter as the main reference. For example, in testing the learning rate parameter, the best learning rate value is used as the basic parameter.

3.1. Accuracy

Based on the results of testing the correct prediction ERNN algorithm on aspects of each sentence can be seen in the following Table 4.

The highest accuracy occurs in the <SERVICE> aspect and the lowest <FOOD> aspect.

Table 4. Prediction tokens aspects.

Aspects	Correct prediction	Accuracy
<FOOD>	306	72.51%
<PRICE>	367	86.97%
<SERVICE>	388	91.94%
<AMBIENCE>	310	73.46%

3.2. F1 score

In this study using weighted macro F1 score in calculating f1 score because the dataset used has an unbalanced label, which is the aspect token that has the label

"NOT SENTIMENT" more. Calculation of the f1 score using the confusion matrix in Table 5 as follow.

Table 5. F1 score all classes.

Class Name	F1 Score	Class member	Prediction	
			True Positive	True Negative
Not sentiment	89%	1400	1202	198
Negative	21%	56	12	44
Positive	56%	232	157	75

Weighted $F1$ score as follow

$$\begin{aligned} Averaged\ F1\ score &= \frac{1}{1688}.\big((1400 * 0.89) + (56 * 0.21) + (232 * 0.56)\big) \\ &= \frac{1397.3159}{1688} \\ &= 0.8278 \end{aligned}$$

So, the overall weighted $F1$ score is 82.78%.

Conclusion

From the results of experiments conducted, this research has succeeded in implemented the Elman Recurrent Neural Network algorithm in the case of ABSA. The results of tests that have carried out show the application of the Elman Recurrent Neural Network algorithm in the case of ABSA produces an average accuracy of 81.22% and f1 score of 82.78%. Tests showed the lowest results in predicting the label "NEGATIVE" which is equal to 21%. This is happening because in tokens are often not found classes in the dictionary (UNKNOWN TOKEN). An example can be seen in the following sentence: "porsi makanannya tidak ada explainnya". The word "explain" is not in the dictionary so it is replacing by UNKNOWN TOKEN which should be predicting on the <FOOD> "NEGATIVE" aspect to "POSITIVE". This happens in sentences that use informal language or a mixture of Indonesian and foreign languages. The imbalance classes make it difficult to predict the label "NEGATIVE" because the gap between the number of “NEGATIVE" label and "NOT SENTIMENT" is too large. This is common because aspects cannot always be containing in a sentence.

References

1. Chaturvedi, I.; Cambria, E.; Welsch, R.E.; and Herrera, F. (2018). Distinguishing between facts and opinions for sentiment analysis: Survey and challenges. *Information Fusion*, 44, 65-77.
2. Katte, T. (2018). Recurrent neural network and its various architecture types. *International Journal of Research and Scientific Innovation*, 5(3), 124-129.
3. Tran, N.M. (2018). Aspect based sentiment analysis using NeuroNER and bidirectional recurrent neural network. *In Proceedings of the Ninth International Symposium on Information and Communication Technology*,1-7.
4. Tarasov, D. (2015). Deep recurrent neural networks for multiple language aspect-based sentiment analysis of user reviews. In Proceedings of

International *Conference of Computational Linguistics and Intellectual Technologies Dialog*, 2, 53-64.

5. Widiastuti, N.I. (2018). Deep learning-now and next in text mining and natural language processing. *IOP Conference Series: Materials Science and Engineering*, Bandung, Indonesia, 407(1), 12114.
6. Liu, P.; Joty, S.; and Meng, H. (2015). Fine-grained opinion mining with recurrent neural networks and word embeddings. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal 1433-1443.
7. Cahyadi, A.; and Khodra, M.L. (2018). Aspect-based sentiment analysis using convolutional neural network and bidirectional long short-term memory. *Proceedings of the 5th International Conference on Advanced Informatics: Concept Theory and Applications* , Krabi, Thailand, 124-129.
8. Uysal, A.K.; and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing and Management*, 50(1), 104-112.
9. Purwarianti, A.; Andhika, A.; Wicaksono, A.F.; Afif, I.; and Ferdian, F. (2016). InaNLP: Indonesia natural language processing toolkit, case study: Complaint tweet classification. *2016 International Conference on Advanced Informatics: Concepts, Theory and Application*, Penang, Malaysia, 1-5.
10. Brun, C.; and Nikoulina, V. (2018). Aspect based sentiment analysis into the wild. *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium, 116-122.
11. Färber, M.; Qurdina, A.; and Ahmedi, L. (2019). Team peter brinkmann at semeval-2019 task 4: Detecting biased news articles using convolutional neural networks. *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minnesota, USA, 1032-1036.
12. Aggarwal, C.C. (2018). *Neural networks and deep learning: A textbook*. Springer publications.
13. Shao, G.; Kobayashi, Y.; and Kishigami, J. (2018). Traditional Japanese haiku generator using RNN language model. *2018 IEEE 7th Global Conference on Consumer Electronics*, Nara, Jaopan, 263-264.
14. Dahl, G.E.; Sainath, T.N.; and Hinton, G.E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, 8609-8613.
15. Botía, J.A.; Álvarez-García, J.A.; Fujinami, K.; Barsocchi, P. and Riedel, T. (2013). *Evaluating AAL systems through competitive benchmarking.* Springer publication.
16. Cong, G.; and Buratti, L. (2018). On Adam trained models and a parallel method to improve the generalization performance. *2018 IEEE / ACM Machine Learning in HPC Environments*, Texas, USA, 85-94.
17. Wibawa, M.S. (2017). Pengaruh fungsi aktivasi, optimisasi dan jumlah epoch terhadap performa jaringan saraf tiruan. *Jurnal Sistem dan Informatika*, 11(2), 167-174.