Deep Learning Model for Sentiment Analysis on Short Informal Texts

Sam Farisa Chaerul Haviana[1], Bagus Satrio Waluyo Poetro[2]
[1,2]Department of Informatics Engineering, Universitas Islam Sultan Agung, Indonesia

Article Info

Article history:

Received Jul 11, 2021
Revised Dec 20, 2021
Accepted Jan 11, 2022

Keyword:

Sentiment Analysis
Deep Learning
Short text

ABSTRACT

This paper proposes a classification model to classify short informal texts. Those short informal texts were texts that were noisy, typos, irregular, and could consist of a very small number of words or even only a single word. The proposed model was trained using a dataset collected from student comments from an application called Evaluasi Dosen Oleh Mahasiswa (EDOM). This application assesses the lecturers using questionnaires filled out by students. It also records the student's comments but is not part of the evaluation calculation, therefore this work makes the data possible to be part of the assessment through sentiment analysis. This work focuses on building suitable preprocessing algorithm and building a simple deep learning network. The preprocessing algorithm was based on multiple word n-gram and Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, and the network was built with a relatively shallow network. To evaluate the model in real usage, an application was built. The results were very convincing, reaching 0.979 in accuracy and 0.63 in F1-Score. Nonetheless, the imbalanced dataset was the only factor that needed to be investigated further for better overall performance.

Corresponding Author:

Sam Farisa Chaerul Haviana,
Department of Informatics Engineering,
Universitas Islam Sultan Agung,
Jalan Kaligawe KM 4, Semarang, Indonesia.
Email: sam@unissula.ac.id

1. INTRODUCTION

Sentiment analysis has become one of highly active research in Natural Language Processing (NLP) and is rapidly increasing the number of methods and their implementations. Many implementations of sentiment analysis use popular machine learning methods, such as support vector machines (SVM), naive Bayes, K-nearest neighbor (K-NN), neural networks, or modifications of these methods. SVM for sentiment analysis, in many reviews, has been domineering and offers better accuracy compared to the other methods [1]–[8]. Table 1 shows a comparison of some research on sentiment analysis with popular traditional machine learning (ML) methods on sentiment analysis. However, SVM is not the only method used in order to get high accuracy and performance in sentiment analysis. Deep learning techniques, on the other hand, cannot be overlooked for their potential in sentiment analysis compared to methods that are already popular like SVM and Naïve Bayes. Deep learning techniques could provide minimal constraints or data to do the task in sentiment analysis [9]. The popularity of deep learning techniques for sentiment analysis has increased recently. Deep Neural Network (DNN), Convolutional Neural Network (CNN), and hybrid approaches have become widely used techniques in sentiment analysis [10]. Nonetheless, the deep learning approach requires fine-tuning of modelling the network and determining the optimal hyper-parameters. A deep learning network also requires optimal depth for the best performance. Deeper deep learning networks did not always lead to better accuracy. It is also harder to train, and may result in decreased accuracy and increased training errors [11].

Table 1. Accuracy Comparison of ML method on sentiment analysis

Study by	Text Preprocessing	Feature Extraction	Classification Method	Accuracy (%)
Yennimar; Rizal, Reyhan Achmad [4]	Tokenize, Filter Token, Stem, Filter Stop Words	TF-IDF, Features Weighting	Naïve Bayes	89
			SVM	**94**
			KNN	67
			ANN	68
Chandani, Vinita; Wahono, Romi Satria; Purwanto [5]	Tokenize, Filter Token, Stem (Porter), Filter Stop Words	Information Gain, Chi-Square, Sequential Forward Selection, Sequential Backward Elimination	ANN	51.8
			SVM	**81.1**
			Naïve Bayes	74
A., Vishal; Sonawane, S.S. [8]	Char Filtering, Remove Stop Words, Expand Acronyms, Rmove Non-English	unigram	Naïve Bayes	74.56
			SVM	**76.68**
			Max Entropy	74.93

Building a deep learning model is very challenging, however in sentiment analysis, deep learning has proven to have outperformed traditional machine learning methods [12]. As shown in Table 2, results of [12], some deep learning approaches acquired better accuracy on some common datasets in sentiment analysis compared to some traditional machine learning methods. Selecting a good machine learning algorithm or building a precise deep learning model is only a part of the process of building a good implementation in sentiment analysis.

Table 2. Accuracy (in %) between Deep Learning and Machine learning methods in Sentiment Analysis [12]

Algorithm	Category	Datasets		
		Twitter tweets	IMDB Reviews	Hotel Reviews
Naïve Bayes	Machine Learning	61.28	71.98	70.9
Logistic Regression	Machine Learning	72.90	85.48	80.12
Random Forests	Machine Learning	72.44	84.98	80.37
LSTM	Deep Learning	74.54	87.58	**81.29**
CNN	Deep Learning	74.44	**88.98**	81.28
CNN + LSTM	Deep Learning	**74.92**	88.28	81.19

Another decisive part of the process, and considered as the most important part, is preprocessing the data. In many implementations and studies, the most common preprocessing steps in sentiment analysis are character filtering (HTML tags filter, non-alphabet filter, non-Unicode filter, etc), case folding, stop words filter, and stemming or lemmatization. Then later building vector of token or building vector of statistical measure like TF-IDF. Those steps of text preprocessing were often used in common datasets like Twitter tweets, the Movie review dataset or Hotel reviews as shown in Table 2. Datasets like Movie and Hotel reviews were considered as long text datasets. The popular Large Movie Review dataset [13] for example, consists of an average of around 130 words, with the minimum number of words is 4 and the maximum number of words is 1460. Some Twitter datasets are an exception, as it can be considered as short text datasets. Many sentiment analysis studies and research have been done for those kinds of long text datasets or documents. Unlike short texts, long texts relatively have more context information or features to extract. On the other hand, short texts can be noisy, lack of context information, and the constituent words are more infrequent, which makes short texts more challenging in sentiment analysis. Commonly used preprocessing and classification methods may not suitable for sentiment analysis on short text. On the short text that already consists of few words, preprocessing like removing stop words could produce an inaccurate message of the text. The application of stemming in preprocessing the text for sentiment analysis is also arguable that stemming could provide better performance in classifying the text. Usage of stemming and stop word removal on sentiment analysis does not significantly provide better accuracy and tends to lower performance [14]. Therefore, this study proposes different preprocessing steps and utilizes deep learning to build sentiment analysis method on short texts with reliable performance.

2. RESEARCH METHOD

The proposed work consists of two main focuses, first, defining the suitable preprocessing algorithm and second, building the deep learning network architecture to achieve a more efficient classifier model for short text sentiment analysis. Then, an application prototype was built to test and validate the model under the running application. This work also investigates how the characteristics of the data set affect the model.

2.1. Datasets

The dataset was built based on student comments in an application called Evaluasi Dosen Oleh Mahasiswa (EDOM) that can be accessed privately at http://edom.unissula.ac.id/. EDOM collects data from questionnaires filled out by students and uses it for lecturer assessments. The questionnaires included the student comments for the lecturers. However, those comments were not part of the calculation of the assessment. This work used the comment texts from a full year of the assessment period in EDOM and resulted in 1792 individual comments, then labeled those comments based on positive, negative, and neutral sentiment to build the dataset. The composition of labeled data is 43.5% positive labeled data, 50.8% negative labeled data, and 5.7% neutral labeled data. This dataset is accessible on Github: https://github.com/samfch/edom-sentiment/tree/main/dataset. Table 3 shows the sample of some labeled student comments in the dataset.

Table 3. Example of labeled student comments in dataset

Student Comments in Bahasa and English meaning	Label
agak lebih pelan kalau menjelaskan bu (a bit slower to explain ma'am)	negative
asik dan menyenangkan (fun and pleasant)	positive
bagus (good)	positive
bagus banget (very good)	positive
hmmm (hmmm)	neutral
jam kuliahnya lebih on time lagi pak (more on time please, sir)	negative
jangan moody an bu (Don't be moody ma'am)	negative

Table 3 shows that those comments are mainly written in irregular and informal Bahasa. However, some comments also contain English words, such as "*on time*" and "*moody*". With that, this work ignores the language factor in the preprocessing as the data could consist of multiple languages. The dataset also consists of some single-word comments such as "*bagus*" (good) and "*hmmm*". Those single and uncommon word comments like "*hmmm*" can obscure the meaning or context of the comment. Even so, those single irregular and uncommon word comments have to be part of the dataset. This leads to another consideration that is not using the stemming and removing stopwords other than already described in [14]. This avoids short text especially text with lesser words from losing its features. Another characteristic of this labeled dataset is the number of typos. Words with typos are very informal, however they still have to be accepted as part of the dataset and labeled with the appropriate label. The approach to handling it was to build a context-free text tokenizer, and this work utilizes n-gram.

2.2. Preprocessing Algorithm

The first focus of this work was to build suitable preprocessing steps for the dataset. Some common text filterings were still used, like HTML stripper and Text Normalizer. HTML stripper was intended to minimize noisy tags or unwanted characters and Text Normalizer was intended to normalize text cases. The proposed preprocessing utilizes a multi N-gram based tokenizer, since N-gram retain the pattern of consecutive words, a multi N-gram could increase the features of a short text. A single word N-gram builds a set of N consecutive words in the text. The number of N-gram from a text containing T words is expressed in formula (1).

$$\textit{Num. of N-gram} = T+1-N \tag{1}$$

Here, T is the total number of words in a text, and N is the number of consecutive words. As the N-gram built from T words, the number of N-gram cannot exceed T since each N-gram defined by every word in the text except the last N-1 words. For example, given the text "three words text", a single word unigram produces T+1–N = 3+1–1 = 3 N-gram, that is "three", "words" and "text". A single bigram of given text produces 2 N-gram, that is "three words" and "words text", and a trigram produces only one N-gram that is the text itself "three words text" as T = N. A single N-gram with T < N will produce no N-gram. Text with only one word will produce no N-gram when applied to an N-gram with N > 1. Considering a single N-gram can produce an empty result of N-gram, this work proposed a multi N-gram approach. Multi N-gram building each N-gram for every incremental N value. If N is in the range of A to B, then the number of A-gram to B-gram of T words can be calculated using formula (2).

$$\textit{Num. of A-gram to B-gram} = (T+1)((B-A)+1) + A(A-1)/2 - B(B+1)/2 \quad (2)$$

In the case of A=B, the number of N-gram produced is the same result of the formula (1). A special case where A=1 and B=T produces a total of all possible N-gram of T words text. That is, the number of possible N-gram can be calculated using formula (3).

$$\textit{Num. of possible N-gram in T words text} = T(T+1)/2 \quad (3)$$

Using multi N-gram produces more N-gram tokens that extract more features from the text. A short text with only a single word produces at least one N-gram value in a multi N-gram where N starts from 1. Because of the lack of features in short texts, maintaining the features of short texts is necessary.

The next step of this proposed algorithm was building the token for all possible N-gram of the training dataset, then taking the Term Frequencies (TF) of words token and applying Inverse Document Frequencies (IDF) to weight them. In the implementation, a pipeline of text preprocessing steps was built before continuing to the network model. Figure 1 shows the pipeline of this work preprocessing steps.

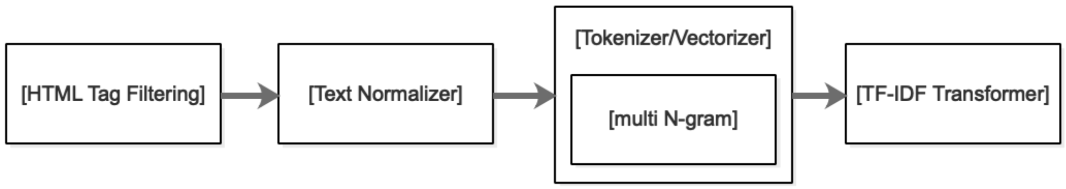

Figure 1. Preprocessing Pipeline

The proposed preprocessing steps were intended to be able to maintain the features of short texts. Removing any word from a short text could lead to context alteration and incorrect sentiment. Therefore, steps like stemming or lemmatization and stopword filtering were considered unnecessary steps in this study.

2.3. Network Architecture

The proposed network was intended to build an effective classifier for sentiment analysis on short texts by determining type of layers, activation functions, and the optimizer. As well as adjusting the hyperparameters. The network built in this work consists of Dense / Fully Connected layers, Leaky ReLU Activation [15], Dropout layers, and a PReLU Activation [16]. Figure 2 shows the network architecture for this work.

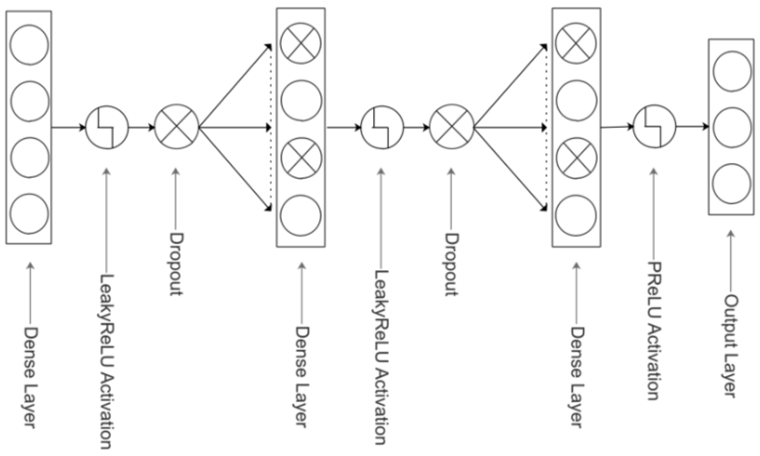

Figure 2. Proposed Network Architecture

The proposed network performs Dropout in every output of the fully connected layer to prevent overfitting [17]. Leaky Rectified Linear Unit (Leaky ReLU) and Parametric Rectified Linear Unit (PReLU) used to achieve lesser errors and better performance as they outperform the original ReLU [18]. Rectified linear unit (ReLU) activation was first used in [19] for image classification using the Restricted Boltzmann Machine (RBM). The ReLU function is defined as (4).

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ 0 & if\ x_i > 0 \end{cases} \tag{4}$$

Here, y_i is the corresponding output of the activation function of the ith input channel, where x_i is the input value of the ith channel. ReLU is sufficient for most deep network architectures, however, some novel modified ReLU like Leaky ReLU and PReLU achieve better performance. The Leaky ReLU function is defined as (5).

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \frac{x_i}{\alpha_i} & if\ x_i < 0 \end{cases} \tag{5}$$

Leaky ReLU sets the α parameter, where α is the amount of leakage, as a proportion of the input. The α is fixed parameter in range 1 to +∞. With a precise α parameter, leaky ReLU performs much better than the original ReLU [18]. In this proposed network, the α of Leaky ReLU activations was set to 0.3. In line with Leaky ReLU, PReLU also improves the classification accuracy [16]. The PReLU function itself is defined as (6).

$$y_i = \begin{cases} x_i & if\ x_i > 0 \\ \alpha_i x_i & if\ x_i \leq 0 \end{cases} \tag{6}$$

In PReLU, α is a learnable parameter controlling the slope of the negative part. When α = 0, PReLU is the same as original ReLU and if α is a fixed and small, PReLU become Leaky ReLU. Leaky ReLU and PReLU handles the negative part by using α. This provides flexibility to set the optimal value on different types of data. Both Leaky ReLU and PReLU confirmed to have better performance than the original ReLU. According to that, this proposed network utilizing Leaky ReLU and PReLU in order to achieve the expected results.
The last step in constructing the proposed deep learning network was choosing the optimizer to achieve better training speed and performance. Here, this work utilizes the Adamax optimizer, as a variant of the Adam optimizer [20]. Adam and Adamax have been very promising, in terms of gaining huge performance and faster training speed. The experiment in [20] reported that Adam and Adamax were straightforward to implement and consumed little memory. We expect the benefit of the Adamax optimizer to be applied to this proposed deep learning network

3. RESULTS AND DISCUSSION

During training, the model was evaluated using Matthews Correlation Coefficient (MCC) and Cross-Entropy loss in every epoch running under the training process. The results for MCC and Cross-Entropy are shown in Figure 3. Referring to the loss values, surprisingly, the model performs very convincingly under the proposed network and selected parameters. Losses are decreasing with each epoch and eventually reach a good value as expected. This experiment yielded a 0.054 loss on the final epoch. On MCC validation, although the model was showing good progress in every epoch, it could be optimized further. The best MCC obtained was 0.446, which is relatively low in general. The proposed model was also implemented as a web-based application and validated under a running state. To get an overview of the model's performance, F1-score and Accuracy were used as the validation metrics. The model was tested by running the application several times and validated it on every run. The number of test data was also randomized in order to observe the stability of the F1 score and Accuracy. The test results are shown in Figure 4.

Both the Accuracy and F1 scores obtained were adequately stable. This shows the model's stability in predicting new data. The model was able to reach high overall accuracy, with the highest overall accuracy score reaching 0.979 (97,9%), although, for the imbalanced labeled dataset, accuracy can be reviewed further. The evaluated F1 scores have an average of 0.63 which is acceptable in terms of the gap with the accuracy score. The result of the F1 scores on average was 0.63, which indicates that the predicted per class accuracy was also imbalanced.

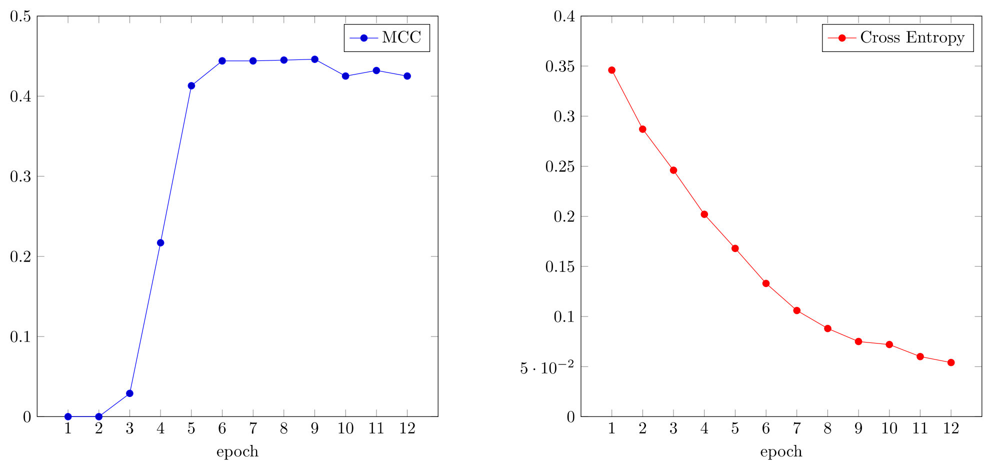

Figure 3. MCC and Cross Entropy during training process

Low accuracy occurs in sentiment classes with a smaller distribution. In this work, neutral sentiment was very rarely predicted in this case.

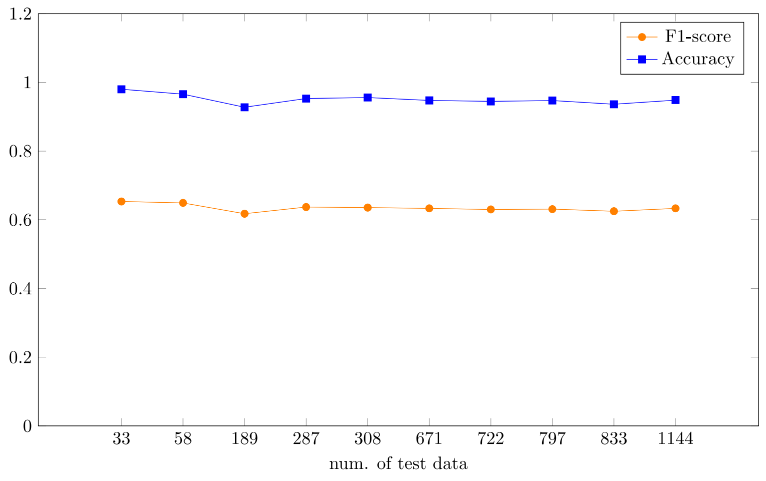

Figure 4. Results for F1 score and Accuracy on running application

The proposed deep learning network outperforms previous works using machine learning approaches such as in [21]–[23]. The highest accuracy was reported in [4] that reach 94% using an SVM classifier. However, that result in general is still lower than this proposed method's accuracy, which reached 0,979 (97,9%). Compared to other deep learning approaches, which is Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) that is already published in [10], the proposed model are very convincing enough as the accuracy also higher than all reported methods and dataset in [10]. Nevertheless, due to the dataset differences, this method still needs to get further testing. In terms of sentiment analysis on short texts, the proposed method was also able to provide a better accuracy than what had been done in [24] that reach 85.5% of accuracy. However, the proposed method is very low in F1 score compared to existing deep learning approaches in [10] and, [24]. The F1 scores could reflect the imbalanced dataset used in this work. Lower F1 score means that this proposed model also had an imbalanced per class accuracy. The neutral class was obviously rarely predictable. This could be due to the lower distribution of the neutral class, compared to the positive and negative classes.

4. CONCLUSION

Based on the validation results, it can be concluded that the model performs very convincingly on short informal text sentiment classification. The model was able to deliver convincing evidence for even classifying single-word texts. The proposed preprocessing was also capable of maintaining features of short texts and works under noisy, short, typo, and informal texts. With relatively small networks, it is also very important to choose the right parameters based on the data characteristics to achieve the expected performance. The only factor necessary to be investigated further is the imbalanced dataset and how it affects the validation results and refracts the classification results in applications.

ACKNOWLEDGEMENTS

Big gratitude to the Research and Community Service Unit of Universitas Islam Sultan Agung (LPPM UNISSULA) for providing the funding. Also, many thanks to Fakultas Teknologi Industri (FTI) members for helping to collect the data used in this study.

REFERENCES

[1] J. Sadhasivam, R. B. Kalivaradhan, and S. Jayavel, "Survey of various algorithms used in twitter for sentiment analysis," *J. Crit. Rev.*, vol. 6, no. 6, pp.449–454, 2019, doi: 10.31838/jcr.06.06.69.

[2] H. Rahmath and T. Ahmad, "Sentiment Analysis Techniques-A Comparative Study," *IJCEM Int. J. Comput. Eng. Manag.*, vol. 17, no. 4, pp.2230–7893, 2014, [Daring]. Available in: www.IJCEM.orgIJCEMwww.ijcem.org.

[3] S. Rani, "Sentiment Analysis: A Survey," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. V, no. VIII, pp. 1957–1963, 2017, doi: 10.22214/ijraset.2017.8276.

[4] Y. Yennimar and R. A. Rizal, "Comparison of Machine Learning Classification Algorithms in Sentiment Analysis Product Review of North Padang Lawas Regency," *SinkrOn*, vol. 4, no. 1, pp. 268, 2019, doi: 10.33395/sinkron.v4i1.10416.

[5] V. Chandani, R. S. Wahono, and P. Purwanto, "Komparasi algoritma klasifikasi Machine Learning dan feature selection pada analisis sentimen review film," *J. Intell. Syst.*, vol. 1, no. 1, pp. 56–60, 2015.

[6] S. Rana and A. Singh, "Comparative analysis of sentiment orientation using SVM and Naive Bayes techniques," 2017, doi: 10.1109/NGCT.2016.7877399.

[7] A. L. Firmino Alves, C. de S. Baptista, A. A. Firmino, M. G. de Oliveira, and A. C. de Paiva, "A Comparison of SVM Versus Naive-Bayes Techniques for Sentiment Analysis in Tweets: A Case Study with the 2013 FIFA Confederations Cup," in *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, 2014, pp. 123–130, doi: 10.1145/2664551.2664561.

[8] V. A. and S. S. Sonawane, "Sentiment Analysis of Twitter Data: A Survey of Techniques," *Int. J. Comput. Appl.*, 2016, doi: 10.5120/ijca2016908625.

[9] P. Singhal and P. Bhattacharyya, "Sentiment Analysis and Deep Learning : A Survey," *CoRR*, 2016.

[10] N. C. Dang, M. N. Moreno-García, and F. De la Prieta, "Sentiment analysis based on deep learning: A comparative study," *Electron.*, vol. 9, no. 3, 2020, doi: 10.3390/electronics9030483.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[12] D. Kansara and V. Sawant, "Comparison of Traditional Machine Learning and Deep Learning Approaches for Sentiment Analysis," in *Advanced Computing Technologies and Applications*, H. Vasudevan, A. Michalas, N. Shekokar, and M. Narvekar, Ed. Singapore: Springer Singapore, 2020, pp. 365–377.

[13] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Jun 2011, pp. 142–150, [Daring]. Available in: http://www.aclweb.org/anthology/P11-1015.

[14] A. W. Pradana and M. Hayaty, "The Effect of Stemming and Removal of Stopwords on the Accuracy of Sentiment Analysis on Indonesian-language Texts," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, no. 3, pp. 375–380, 2019, doi: 10.22219/kinetik.v4i4.912.

[15] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *ICML Work. Deep Learn. Audio, Speech Lang. Process.*, vol. 28, 2013.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1026–1034, 2015, doi: 10.1109/ICCV.2015.123.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014, [Daring]. Available in: http://jmlr.org/papers/v15/srivastava14a.html.

[18] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," 2015, [Daring]. Available in: http://arxiv.org/abs/1505.00853.

[19] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair," in *Proceedings of ICML*, 2010, vol. 27, pp. 807–814.

[20] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[21] Y. Yunitasari, A. Musdholifah, and A. K. Sari, "Sarcasm Detection For Sentiment Analysis in Indonesian Tweets," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 13, no. 1, pp. 53, 2019, doi: 10.22146/ijccs.41136.

[22] M. Z. Naf'an, A. A. Bimantara, A. Larasati, E. M. Risondang, and N. A. S. Nugraha, "Sentiment Analysis of Cyberbullying on Instagram User Comments," *J. Data Sci. Its Appl.*, vol. 2, no. 1, pp. 88–98, 2019, doi: 10.21108/jdsa.2019.2.20.

[23] H. A. Santoso, E. H. Rachmawanto, A. Nugraha, A. A. Nugroho, D. R. I. M. Setiadi, and R. S. Basuki, "Hoax classification and sentiment analysis of Indonesian news using Naive Bayes optimization," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 2, pp. 799–806, 2020, doi: 10.12928/TELKOMNIKA.V18I2.14744.

[24] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *J. Artif. Intell. Res.*, vol. 50, pp. 723–762, 2014, doi: 10.1613/jair.4272.

BIOGRAPHIES OF AUTHORS

Sam Farisa Chaerul Haviana received the Bachelor degree in Informatics from Universitas Islam Sultan Agung in 2009 and received the Master degree from Universitas Diponegoro in Information System in 2014. Currently working as a lecturer at Universitas Islam Sultan Agung. His research interests include machine learning, data mining, image processing, and decision support system applications. He has also been involved in the development of the Science and Technology Index (SINTA) of the Ministry of Research and Technology of Indonesia since 2017.

Bagus Satrio Waluyo Poetro received the Bachelor degree in Informatics from Diponegoro University in 2010 and received the Master degree from Gadjah Mada University in Computer Science in 2013. Now currently working as a lecturer at Universitas Islam Sultan Agung. His research interests include Computer Security, Image Processing, Machine Learning and also Multimedia Technology. He also involved in some Start-Up business projects.