Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 3, June 2024, pp. 1913~1919
ISSN: 2302-9285, DOI: 10.11591/eei.v13i3.6041

Classifying possible hate speech from text with deep learning and ensemble on embedding method

Ebenhaiser Jonathan Caprisiano, Muhammad Hafizh Ramadhansyah, Amalia Zahra
Department of Computer Science, BINUS Graduate Program, Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article Info

Article history:

Received Feb 18, 2023
Revised Jun 28, 2023
Accepted Sep 28, 2023

Keywords:

Ensemble method
Global vector
Hate speech
Long short-term memory
Text classifying
Term frequency times inverse document frequency
Word to vector

ABSTRACT

Hate speech can be defined as the use of language to express hatred towards another party. Twitter is one of the most widely used social media platforms in the community. In addition to submitting user-generated content, other users can provide feedback through comments. There are several users who intentionally or unintentionally provide negative comments. Even though there are regulations regarding the prohibition of hate speech, there are still those who make negative comments. Using the deep learning method with the long short-term memory (LSTM) model, a classifier of possible hate speech from messages on Twitter is carried out. With the ensemble method, term frequency times inverse document frequency (TF-IDF) and global vector (GloVe) get 86% accuracy, better than the stand-alone word to vector (Word2Vec) method, which only gets 80%. From these results, it can be concluded that the ensemble method can improve accuracy compared to only using the stand-alone method. Ensemble methods can also improve the performance of deep learning systems and produce better results than using only one method.

Corresponding Author:

Ebenhaiser Jonathan Caprisiano
Department of Computer Science, BINUS Graduate Program, Master of Computer Science
Bina Nusantara University
Jakarta, 11480, Indonesia
Email: ebenhaiser.caprisiano@binus.ac.id

1. INTRODUCTION

Hate speech can be defined as the use of language to express hatred for a group or to humiliate person or members of that group [1]. Hate speech can also be defined as a deliberate attack directed at a particular group motivated by aspects of group identity [2]. In this technology era, internet and social media platforms have become an integral part of social life [3]. Twitter is one of the social media platforms that is widely used by the public. Twitter is a social media service that allows users to share content such as text, images, and videos, and users can also view content from other users. Apart from just viewing content, users can also provide feedback regarding the content in the form of comments. However, the comments given by the public can be in the form of positive or negative comments. Many users use social media to spread hate speech that is motivated by personal desires. But there are also some users who accidentally leave negative comments or don't know the meaning of the message. There is a possibility that the writer does not mean to write negatively, but the reader can interpret the writing negatively.

There is debate about banning hate speech. According to a written journal article by Howard [4], this debate should be divided into several parts. The first concerns the scope of the moral right to freedom of expression. Second, there is the moral obligation to refrain from hate speech. The third relies on pragmatic concerns involving epistemic fallibility, abuse of state power, and the benefits of speech against coercion.

Now, there are regulations regarding hate speech on social media. However, it is not uncommon for users to still provide negative comments or hate speech.

So, with this, research will be carried out to classify possible hate speech from text using deep learning with two layers, namely the embedding layer and the layer for long short-term memory (LSTM). The embedding layer will use word to vector (Word2Vec) and the ensemble methods of term frequency times inverse document frequency (TF-IDF) and global vector (GloVe). The LSTM was selected in this study because it performs effectively on a wide range of large issues and is now frequently used [5]. Previously, research on emotion detection using TF-IDF and the LSTM model by Haryadi and Kusuma [6] found an accuracy of 99.22% using LSTM and 99.18% using nested LSTM. Then, research conducted by Nurrohmat and Azhari [7] found that a sentiment analysis of the novel had an accuracy of 72.85% using Word2Vec with LSTM. As quoted from research on hate speech detection by Malik *et al.* [8], using GloVe with the Bi-LSTM model obtains 84% accuracy in the first dataset.

The problem that will be worked on in this paper is to compare the level of accuracy of the results of classifying hate speech between Word2Vec and the ensemble method from TF-IDF and GloVe. This study is expected to be useful in identifying text messages that may contain hate speech. With this, it can also help readers decide the intent of text messages that are likely to contain hate speech. It is also hoped that the resulting data from this study can be used as a basis for future research and the development of applications that aim to detect hate speech.

2. METHOD

The following is a schematic of the stages in the hate speech classification method, as shown in Figure 1. These are the stages that used in this research. In the preprocessing phase, the data undergoes several steps such as tokenization, stop word removal, and lemmatization. Once the preprocessing is complete, the next step is embedding. The embedding methods employed will be Word2Vec, TF-IDF, and Glove. Word2Vec will be implemented as a standalone model, while TF-IDF and GloVe will be used in an ensemble approach. Subsequently, the data will be trained using an LSTM, and the results will be discussed.

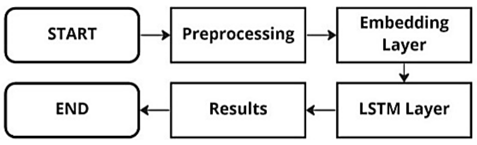

Figure 1. Hate speech classification flowchart

Based on Figure 1, there are steps that are divided into: i) preprocessing, ii) embedding layers, iii) LSTM layer, and iv) results. These steps will be elaborated on in the following sections. The results obtained will include performance metrics such as precision, recall, F1-score, and accuracy. The outcomes of this research will be compared to draw conclusions about the model that exhibits the highest efficiency and accuracy.

2.1. Preprocessing

In preprocessing, there are several steps, namely: i) tokenization, ii) stop word removal, and iii) lemmatization. Tokenization is the process of breaking down text into words, phrases, symbols, and other elements [9]. It aims to be able to explore the words in one sentence. The stop words referred to here are some words that are often found but are meaningless because they are only used to combine words in sentences [9]. For example, "it," "and," "the," and so on make a bad index on a document [10]. The next step is lemmatization. Lemmatization is similar to stemming, which breaks down words into root words. However, lemmatization parses words according to the context in which they are used because there are several words that have multiple meanings [11]. So, in this preprocessing step, we are sorting and filtering sentences so they can be read by computer algorithms.

The dataset used was obtained from the Kaggle website, which contains tweets about the grand old party (GOP) Debate in Ohio in early August 2016 [12]. This dataset containe index, id, candidate, candidate_confidence, relevant_yn, relevant_yn_confidence, sentiment, sentiment_confidence, subject_matter, subject_matter_confidence, candidate_gold, name, relevant_yn_gold, retweet_count, sentiment_gold, subject_matter_gold, text, tweet_coord, tweet_created, tweet_id, tweet_location. This dataset contains 13871 total indexes. The sentiment column contains 2236 positive, 8493 negative, and 3142 neutral. Because the main objective of this study is to classify positive and negative data, neutral data will be deleted, so that the number of datasets that will be used is 10729. The diagram in Figure 2 shows the sum of both positive and negative data.

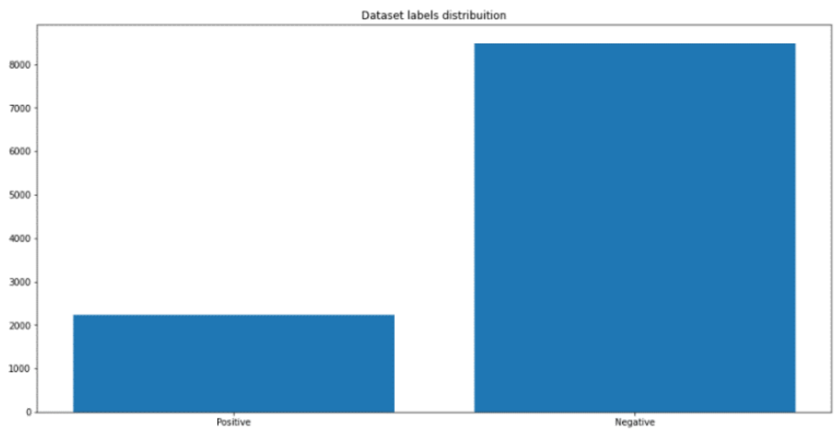

Figure 2. Positive and negative dataset diagram

2.2. Embedding layer

Word embedding is a vector representation of words obtained by inserting words with semantic and syntactic meanings obtained from a large corpus [13]. In this embedding layer, three techniques are used, namely the stand-alone Word2Vec and the ensemble method. The methods are going to be ensembled is the TF-IDF and GloVe models.

2.2.1. Word to vector

Word2Vec is a neural network that represents words in vector form [14]. Word2Vec has two models, namely the continuous bag-of-words (CBOW) model and the continuous skip-gram model. The CBOW model is used in this study. The CBOW model combines existing words to predict middle words [15]. Figure 3 shows the layers in the CBOW architecture. The CBOW layer projects all words to the same position, and the all-word vector maintains the mean and shares the positions of all words [16]. The CBOW architecture predicts current words based on sentence context [17].

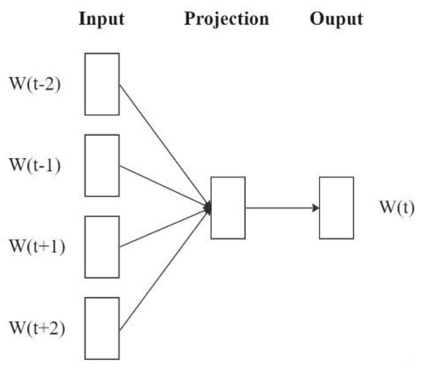

Figure 3. CBOW architecture [15]

2.2.2. Ensemble method

The ensemble method improves the prediction performance of a single model by training many models and combining predictions from these models [18]. In this ensemble method, a combination of TF-IDF and GloVe will be used. The reason for using this ensemble method is hoping that can give better results.

2.2.2.1. TF-IDF

TF-IDF is a formal measure that is concentrated into relatively few documents regarding the occurrence and importance of certain words [19]. In other words, TF-IDF counts the number of times a word appears in a document. This is useful for information retrieval and text mining.

$$TF.IDF = TF(t,d)\ x\ IDF(t) \quad (1)$$

TF-IDF is a merger between term frequency (TF) and inverse document frequency (IDF). TF is the number of times a particular word appears in a document [20]. IDF is the calculation of the number of documents in a collection that contain the term in question [21].

2.2.2.2. GloVe

The last embedding technique that will be used together with TF-IDF in the ensemble method is GloVe. Based on literature review, the GloVe word embedding model has a high level of accuracy compared to other word embedding models such as FastText and others. GloVe model was first introduced in 2014 and is quite popular at this time. GloVe is a model that captures global corpus statistics, which are the main source of information for studying word representation [22]. GloVe, in other words, calculates the relationship between words in a text based on the frequency with which the word appears. With GloVe, machines can use large datasets with billions of words that may not be accessible to derive statistically strong word meanings [8]. In other words, GloVe is a word fusion method that has advantages in capturing global context, handling proportionality, good interpretability, and reliability in various natural language processing tasks. GloVe's word representation reflects semantic and syntactic relationships, making it a popular and effective choice in a variety of natural language processing applications.

2.3. Long short-term memory layer

For the classification of hate speech from text, the LSTM model was chosen over the transformer-based architecture. The capacity of the LSTM to capture long-term dependencies in sequential data, the restrictions of available computational resources, the incorporation of ensemble methods, and the necessity for interpretability all contributed to this conclusion. This decision was made after careful assessment of the task's nature, available resources, and empirical evaluation to arrive at the best solution in the specific context. When compared to transformer-based architectures, which have numerous models for their individual demands based on the language used in the text, LSTM can cover everything, and the LSTM model continues to be commonly used in several contemporary studies. LSTM is an recurrent neural network (RNN) architecture specifically designed to overcome the problem of missing gradients [23]. The main difference with RNNs is that they have problems with inputs that are too far in the past. Block memory is the hidden layer of the LSTM, which consists of subnets that are connected repeatedly [23]. The LSTM memory block has three gates: i) the input gate (I_t), ii) the forget gate (f_t), and iii) the output gate (f_o) as shown in Figure 4.

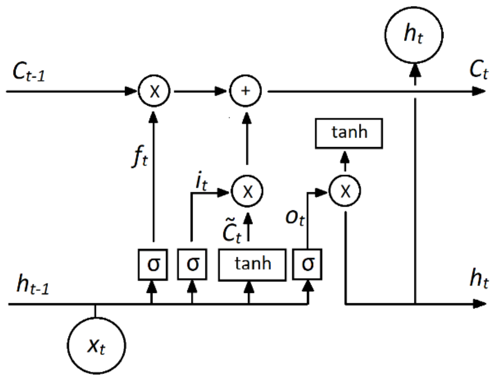

Figure 4. Block memory LSTM [24]

The input gate controls the flow of input activity to memory cells. The output gate controls the flow of output activity to the next network. Before entering it as input via the cell connection, the forget gate measures the state of the internal cell [25]. With this, it can adaptively forget or reset memory cells.

Figure 4 [24] is an example of a block memory. Information is carried through the cell state (C_t), which is like the internal state in the LSTM. In the cell state, there is data that can be discarded or continued to the next memory block and processed in the hidden state.

$$f_t \;=\; \sigma(W_f\;[h_{t-1}, x_t] + b_f) \tag{2}$$

$$C_t = f_t\; x\; C_{t-1} + i_t\; x\; C_t \tag{3}$$

Forget gate (f_t) is calculated by combining the current input (x_t) and the previous hidden state (h_{t-1}), with weight (W_f) and bias (bf), and entering the sigmoid function (σ). The sigmoid function has an output range of 0–1. Then, the forget gate (f_t) is multiplied by the previous cell state (C_{t-i}). So, forget gate is a calculation of how much of the previous cell state is removed. If forget gate is 0, then the previous cell state is deleted vice versa [24].

$$I_t = \sigma(W_i\;[h_{t-1}, x_t] + b_t) \tag{4}$$

$$C_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{5}$$

The input gate (I_t) is similar to the forget gate, consisting of the current input (x_t) and the previous hidden state (h_{t-1}), with weight (W_i) and bias (b_i), and entering the sigmoid function (σ). For input, use weight (W_c) and bias (b_c). Then the results of the input gate and input ($\tilde{C}_t$) are multiplied to enter the cell state (C_t). So the input gate indicates how much input goes into the cell state [24].

$$f_o = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = f_o\; x\; tanh(C_t) \tag{7}$$

The gate output (f_o) consists of the current input (x_t) and the previous hidden state (h_{t-1}), with weight (W_o) and bias (b_o), and enters the sigmoid function (σ). Output (h_t) is the multiplication of the output gate and the hyperbolic tangent (tanh) of the cell state that has been found. So how much output becomes the hidden state (h_t) is regulated by the output gate [24].

3. RESULTS AND DISCUSSION

In this study, this experiment implemented the model using Scikit-Learn, Keras, and NLTK. This research uses a dataset that contains messages sent by the public through the social media application Twitter, which contains tweets about the GOP Debate in Ohio in early August 2016 [12]. The stand-alone classifier using the Word2Vec embedding method uses the categorical_crossentropy for the loss function. The LSTM model uses l layer, 196 neurons and 1 dense. Total parameters that are used in this model are 3,333,445. This model gets 40% precision, 50% recall, and a 44% F1-score on the macro average. The weighted average gets 64% precision, 80% recall, and a 71% F1-score. Then, in the TF-IDF and GloVe ensemble methods, we also use the LSTM model to get 81% precision, 70% recall, and a 73% F1-score in the macro average. The weighted average gets 85% precision, 86% recall, and an F1-score of 84%.

All macro and weighted average results show that TF-IDF+GloVe outperforms Word2Vec, as do both accuracy results. This model uses binary_crossetropy for the loss function, 196 neurons on the LSTM model. There are 3 dense layers in this model, 21 for the first layer, 21 for the second layer, and 1 for the third layer. Total parameters that are used in this model are 1,063,951. This model has an accuracy of 86%, which is higher than Word2Vec's accuracy of 80%. The results of the two experiments show that the results of the TF-IDF and GloVe ensemble methods can improve accuracy compared to Word2Vec. The results with the ensemble method also show an increase in accuracy over the results obtained by Montalvo, with an accuracy of 85% [12]. The results are shown in Table 1.

Table 1. Macro and weighted average F_1 score performance

Embedding	Model	Macro AVG			Weighted AVG			Accuracy
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	
Word2Vec	LSTM	0.40	0.50	0.44	0.64	0.80	0.71	0.80
TF-IDF+GloVe	LSTM	0.81	0.70	0.73	0.85	0.86	0.84	0.86

4. CONCLUSION

From this experiment, it can be concluded that the ensemble method can improve accuracy and be better than the stand-alone method. This is evident from the experimental results, which show that the accuracy of the ensemble method with TF-IDF and GloVe is better than just one method, namely Word2Vec. It can be seen that the ensemble method should be implemented because, with many methods, it can improve the deep learning system to be better than using only one method and provide more optimal results.

In the future work, based on the experimental results that have been obtained, efforts will be made to optimize accuracy. Then, try to add some other deep learning models in future research. In the ensemble method, other algorithms will be added, such as random forest, logistic regression, and support vector classifier, to find other possible accuracy levels that are more optimal. Other features from Word2Vec, TF-IDF, and GloVe will also be used in future research.

ACKNOWLEDGEMENTS

The author acknowledges the support and resources provided by Bina Nusantara University. Gratitude is also extended to the dataset provider for their contribution to this research. Furthermore, the author appreciates the valuable guidance offered by the journal staff, which has contributed to the improvement of this paper.

REFERENCES

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017*, vol. 11, no. 1, pp. 512–515, May 2017, doi: 10.1609/icwsm.v11i1.14955.

[2] O. de Gibert, N. Perez, A. García-Pablos, and M. Cuadros, “Hate speech dataset from a white supremacy forum,” in *2nd Workshop on Abusive Language Online - Proceedings of the Workshop, co-located with EMNLP 2018*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 11–20, doi: 10.18653/v1/w18-5102.

[3] A. K. Yadav, M. Kumar, A. Kumar, Shivani, Kusum, and D. Yadav, “Hate speech recognition in multilingual text: hinglish documents,” *International Journal of Information Technology (Singapore)*, vol. 15, no. 3, pp. 1319–1331, Mar. 2023, doi: 10.1007/s41870-023-01211-z.

[4] J. W. Howard, “Free speech and hate speech,” *Annual Review of Political Science*, vol. 22, no. 1, pp. 93–109, May 2019, doi: 10.1146/annurev-polisci-051517-012343.

[5] D. G. S. N. Murthy, S. R. Allu, B. Andhavarapu, and M. B. M. Bagadi, “Text based sentiment analysis using LSTM,” *International Journal of Engineering Research and*, vol. 9, no. 5, pp. 1–5, May 2020, doi: 10.17577/ijertv9is050290.

[6] D. Haryadi and G. P. Kusuma, “Emotion detection in text using nested long short-term memory,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, pp. 351–357, 2019, doi: 10.14569/ijacsa.2019.0100645.

[7] M. A. Nurrohmat and S. Azhari, “Sentiment analysis of novel review using long short-term memory method,” *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 13, no. 3, pp. 209–218, Jul. 2019, doi: 10.22146/ijccs.41236.

[8] J. S. Malik, G. Pang, and A. van den Hengel, “Deep learning for hate speech detection: a comparative study,” *arXiv preprint*, 2022, doi: 10.48550/arXiv.2202.09517.

[9] V. Gurusamy and Kannan S, “Preprocessing techniques for text mining,” *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2014.

[10] Z. Yao and C. Ze-Wen, “Research on the construction and filter method of stop-word list in text preprocessing,” in *Proceedings - 4th International Conference on Intelligent Computation Technology and Automation, ICICTA 2011*, IEEE, Mar. 2011, pp. 217–221, doi: 10.1109/ICICTA.2011.64.

[11] M. Anandarajan, C. Hill, and T. Nolan, “Text preprocessing,” in *Handbook of Natural Language Processing, Second Edition*, 2019, pp. 45–59, doi: 10.1007/978-3-319-95663-3_4.

[12] I. Montalvo, “LSTM sentiment analysis | keras,” *Kaggle*, 2022. https://www.kaggle.com/code/italomontalvo/lstm-sentiment-analysis-keras/data (accessed Dec. 07, 2022).

[13] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. J. Kuo, “Evaluating word embedding models: Methods and experimental results,” *APSIPA Transactions on Signal and Information Processing*, vol. 8, no. 1, pp. 1–14, 2019, doi: 10.1017/ATSIP.2019.12.

[14] D. E. Cahyani and I. Patasik, “Performance comparison of tf-idf and word2vec models for emotion text classification,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2780–2788, Oct. 2021, doi: 10.11591/eei.v10i5.3157.

[15] T. Mikolov, Q. V. Le, and I. Sutskever, “Exploiting similarities among languages for machine translation,” *arXiv preprint*, 2013, doi: 10.48550/arXiv.1309.4168.

[16] B. Jang, I. Kim, and J. W. Kim, “Word2vec convolutional neural networks for classification of news articles and tweets,” *PLoS ONE*, vol. 14, no. 8, p. e0220976, Aug. 2019, doi: 10.1371/journal.pone.0220976.

[17] D. Jatnika, M. A. Bijaksana, and A. A. Suryani, “Word2vec model analysis for semantic similarities in english words,” *Procedia Computer Science*, vol. 157, pp. 160–167, 2019, doi: 10.1016/j.procs.2019.08.153.

[18] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, Jul. 2018, doi: 10.1002/widm.1249.

[19] M. Manic, “Data Mining,” in *The Industrial Electronics Handbook - Five Volume Set*, Cambridge University Press, 2011, pp. 1–17, doi: 10.1017/CBO9781139058452.002.

[20] N. Azam and J. Yao, “Comparison of term frequency and document frequency based feature selection metrics in text categorization,” *Expert Systems with Applications*, vol. 39, no. 5, pp. 4760–4768, Apr. 2012, doi: 10.1016/j.eswa.2011.09.160.

[21] S. Robertson, “Understanding inverse document frequency: On theoretical arguments for IDF,” *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, Oct. 2004, doi: 10.1108/00220410410560582.

[22] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” in *EMNLP 2014 - 2014*

Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543, doi: 10.3115/v1/d14-1162.

[23] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2009, doi: 10.1109/TPAMI.2008.137.

[24] C. Olah, "Understanding LSTM Networks," 2015, [Online]. Avaiable: https://colah.github.io/posts/2015-08-Understanding-LSTMs, date accessed May 17, 2023.

[25] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, ISCA: ISCA, Sep. 2014, pp. 338–342, doi: 10.21437/interspeech.2014-80.

BIOGRAPHIES OF AUTHORS

Ebenhaiser Jonathan Caprisiano is a student in the Master of Computer Science, Bina Nusantara University, Indonesia. He studied to get a Bachelor degree in Computer Science from 2019. He also took part in the Bina Nusantara University Master Track program, to get his master degree. His research interests are in text classification, emotion detection, and text mining. Additionally, he has interests in natural language processing (NLP) and machine learning. He can be contacted at email: ebenhaiser.caprisiano@binus.ac.id.

Muhammad Hafizh Ramadhansyah is a student at the Master of Computer Science, Bina Nusantara University, Indonesia. He entered his college in 2019. He also took part in the Bina Nusantara University Master Track Program, to get his master degree. His research interests are in machine learning, text mining, and text classification. Additionally, he also interests in natural language processing (NLP) and deep learning. He can be contacted at email: muhammad.ramadhansyah@binus.ac.id.

Amalia Zahra is a lecturer at the Master of Computer Science, Bina Nusantara University, Indonesia. She received her Bachelor degree in Computer Science from the Faculty of Computer Science, University of Indonesia (UI) in 2008. She does not have a master degree. Her PhD was obtained from the School of Computer Science and Informatics, University College Dublin (UCD), Ireland in 2014. Her research interests cover various fields in speech technology, such as speech recognition, spoken language identification, speaker verification, and speech emotion recognition. Additionally, she also has interest in natural language processing (NLP), computational linguistics, and machine learning. She can be contacted at email: amalia.zahra@binus.edu.