TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 4, August 2021, pp. 1234~1241
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i4.20369

Enhancing text classification performance by preprocessing misspelled words in Indonesian language

Reza Setiabudi, Ni Made Satvika Iswari, Andre Rusli
Department of Informatics, Universitas Multimedia Nusantara, Indonesia

Article Info

Article history:

Received Aug 31. 2020
Revised Jan 11, 2021
Accepted Jan 20, 2021

Keywords:

Indonesian language
Levenshtein distance
Text classification
Typo correction
User feedback

ABSTRACT

Supervised learning using shallow machine learning methods is still a popular method in processing text, despite the rapidly advancing sector of unsupervised methodologies using deep learning. Supervised text classification for application user feedback sentiments in Indonesian Language is one of the applications which is quite popular in both the research community and industry. However, due to the nature of shallow machine learning approaches, various text preprocessing techniques are required to clean the input data. This research aims to implement and evaluate the role of Levenshtein distance algorithm in detecting and preprocessing misspelled words in Indonesian language, before the text data is then used to train a user feedback sentiment classification model using multinomial Naïve Bayes. This research experimented with various evaluation scenarios, and found that preprocessing misspelled words in Indonesian language using the Levenshtein distance algorithm could be useful and showed a promising 8.2% increase on the accuracy of the model's ability to classify user feedback text according to their sentiments.

Corresponding Author:

Andre Rusli
Department of Informatics
Universitas Multimedia Nusantara
UMN Campus, Scientia Boulevard St., Gading, Serpong, Tangerang, Banten 15811, Indonesia
Email: andrerusli19@gmail.com

1. INTRODUCTION

Nowadays, thanks to the rapidly advancing technologies in various sectors, end-user feedback can be gathered and collected easily via various channels such as digital user surveys, social media and mobile application stores. Analyzing these end-user feedback could provide useful information regarding product requirements which could benefit the owners and engineers of a certain product, as the users themselves could participate in determining the product requirements through the feedback they gave. User involvement in software requirements engineering [1], [2] is crucial in delivering a right product. Not only before and during the software development process takes place, user involvement after the product is delivered is also crucial. Developers and product owners would often gather feedback from their end-users to continuously improve their products, thus, making user feedback as one of the most crucial sources of information in order to make a product with the best quality. This is a fundamental aspect in some recent concepts of requirements engineering such as the market driven requirements engineering (MODRE) [3] and the CrowdRE paradigm [4], which gives end-users a prominent role with respect to other stakeholders. In order to analyze and process feedback into useful information, one important point will be the characterization of the feedback properties that may be actually used in the automated prioritization steps [5] and those that need human competences and intervention to be exploited in the decision process [6].

Recent advances in machine learning and natural language processing give way to enable software developers to better process these user feedback automatically, as processing all the collected user feedback text would take tremendous time and effort. However, user feedback in the form of app reviews poses some challenges to build a machine to be automatically processed into useful information. On the one side, they share properties with Tweets and other social media texts, e.g., comparably short and informal language [7], thus making it a challenges for the machine which is often used to dealing with structured data. On the other side, they are similar to product reviews from other domains or platforms, e.g., reviews about household appliances or books on Amazon, as they typically describe the user's opinion about specific aspects [8]. Many works of research have been tackling these challenges in automatically processing text for application user feedback.

Text classification is one branch of natural language processing that is often used in processing feedback texts. Many supervised machine learning methods, such as multinomial Naïve Bayes, logistics regression, and support vector machine, are proven to be effective and applicable in classifying texts in English [9], [10]. However, as a language with less resource compared to English, research in processing Indonesian language is very crucial, especially if we consider the huge number of social media and mobile application users in the country. Many researchers in the field have already conducted experiments in text classification, especially sentiment analysis/classification, in Indonesian language [11]-[15]. Based on results from related works of research, our previous works implemented and evaluated the performance and applicability of several methods in classifying text written in Indonesian language, such as Naïve Bayes algorithm to classify application user feedback based on their sentiments and specific feedback categories [16], [17], multilayer perceptron to identify fake news [18], and a comparison of several shallow machine learning methods to classify user feedback based on their categories [19]. In these various works of research in Indonesian language, many methodologies in classifying texts have been experimented, Naïve Bayes being one of the most popular methods with promising performance as a supervised machine learning classification method. However, as shown in [20], [21], text preprocessing and language normalization techniques, such as preprocessing misspelled words, have shown a huge increase in performance when performing text classification in English. Another paper [22] also shows the potential of Naïve Bayes to be further improved for text classification.

Based on the results of previous works of research, the main contributions of our research could be decomposed into three steps.

- Firstly, our research aims to implement a method for processing misspelled words in Indonesian language using the Levenshtein distance algorithm, which has been proven to search for similar words [23] in the text preprocessing phase.
- Secondly, two classification models using Naïve Bayes algorithm are built, one of them utilizes the method for preprocessing misspelled words and the other does not.
- Finally, we test the both ability of Levenshtein distance in processing misspelled words in the dataset and the ability of the models to classify texts from the test set. The performance results are then compared and analyzed to evaluate the impact of incorporating an additional text preprocessing step when building a text classification model for texts written in Indonesian language.

In section 2, the methodologies used in our reseach and experimentation are described. Section 3 explains the results and analysis of our experimentation, to serve as an evaluation of the methods used in our paper so future works could get some insights based on our works. Lastly, section 4 concludes our paper and mentions several improvements that have not been covered in this paper for future works.

2. RESEARCH METHOD

In conducting the research to implement and evaluate Levenshtein distance algorithm to preprocess misspelled words and multinomial Naïve Bayes to classify user feedbacks, several research steps are done. Figure 1 shows the overall step-by-step research metholodogies that were conducted. The research was started by reviewing literatures and recent works on similar topics, including requirements engineering, user feedback processing, text classification, typo correction, and common evaluation metrics to measure the performance of our proposed model.

Research problems are then elicited and defined after reviewing various literatures. This research aims to implement the Levenshtein distance algorithm to preprocess texts containing misspelled words in Indonesian language which are then used to train our feedback sentiment classification model using multinomial Naïve Bayes. The Levenshtein distance algorithm works by calculating similarities between words to normalize unknown words which are similar to existing words in Indonesian language, thus transforming words considered to be misspelled to words in the Indonesian dictionary. The formula below explains how the

Levenshtein distance algorithm works, here, the distance between two words is defined as the minimum steps required to transform the source string to the target string, via operations such as deletion, insertion, and/or substitution.

$$lev_{a,b}(i,j)f(x) = \begin{cases} \max(i,j) & if \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \quad (1)$$

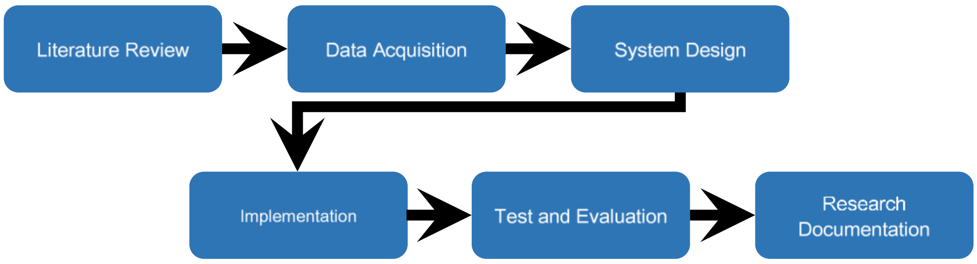


Figure 1. Step-by-step research methodologies

Another formula that is used in our work is to normalize the result of Levenshtein distance calculation which is sensitive to the varying length of the compared strings. We normalize the result using the (2) [24], where length is the maximum length of the two strings that are compared, and distance is the minimum value calculated from (1);

$$score = \frac{length-distance}{length} \quad (2)$$

Furthermore, to evaluate the effectivity of employing the algorithm in text classification, this research also aims to design several test scenarios and measure the accuracy of both the typo correction algorithm and the newly enhanced accuracy of our classification model. In the data acquisition phase, we gathered data from several sources for our experiments. Firstly, in order to correct supposebly misspelled words, we use dictionaries from two sources, one is a digital dictionary of Indonesian language [25] and the other one contains the list of slang words in Indonesian language [26]. Secondly, we collected a portion of the feedback data from previous work which can be found in [11] for training and testing the classification model. The feedback data will be used to train two models, the first one is the one without typo correction and the other implements the typo correction preprocessing step using Levenshtein algorithm. We use two sentiment labels from the dataset, which tell us whether a feedback text is classified as a positive and negative feedback. Below are some examples of end-user feedback consisted in the dataset [11] that were used in our experiment:

a. Positive feedback (241 data):
  - *"aplikasi bagus untuk majalah, buku yg original bikinan Indonesia."*
    (in English: "cool, very useful for original Indonesian magazines and books.")
  - *"keren bro. membantu dalam proses belajar mengajar thank you."*
    (in English: "cool bro. very helpful for teaching and learning thank you.")

b. Negative feedback (241 data):
  - *"selalu crash setiap mau mencari kata!! jangan update ke versi ini!!!"*
    (in English: "it always crashes every time I tried to find a word!! don't update to this version!!!")
  - *"betulin dong aplikasinya, rusak mulu nih!!!!"*
    (in English: "fix the app, it's broken!!!!")

After the required data are gathered, we started to design the system which would later implement all the required natural language processing techniques to fulfill our research objectives. Figure 2 shows the overall flow of our system's program. Firstly, we imported all the required data into the system. We used Python and Jupyter Notebook as our main development environment. After all data has been imported, the system then proceeds to the data cleansing phase. In this phase, the system will conduct common data preprocessing chores such as casefolding to change text data to lowercase, removing non-alphanumeric characters, tokenization, and stemming. The result of the stemming process is a formal reprentstation of a word in Indonesian language. After the data cleansing process is done, the system will move on to the next step which is the main empashize of our research.

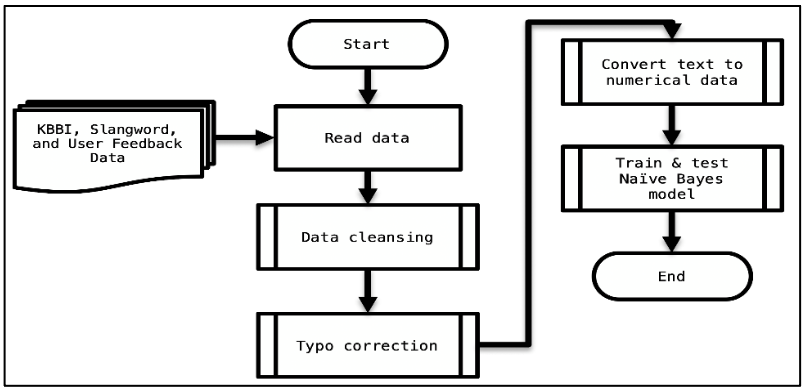

Start
KBBI, Slangword, and User Feedback Data
Read data
Data cleansing
Typo correction
Convert text to numerical data
Train & test Naïve Bayes model
End

Figure 2. Overall system flowchart

Figure 3 below depicts the flowchart of the typo correction process in our system. After initializing some variables, the system will initially check words form the feedback list and match them with both the slang-word list and the Indonesian dictionary list. In our experiment, only words with a specified length from the list will be used for comparison, we used words with lengths plus minus 3 character compared to the input word. Then, if a word is not a member of the slang-word and dictionary list, then the word is considered as unknown and will proceed to the typo correction phase using the Levenshtein distance algorithm. The unknown word will then used as an input and the algorithm will calculate the word distance with every word within both the dictionary and the slangword. Finally, the distance scores are converted into floating point to be normalized [24], and the word with the highest similarity score (minimum distance) is then selected as the correct spelling of the unknown word.

After the text data are cleaned and misspelled words are converted into known words from the dictionary or the slang-word list, as also shown in Figure 2, the system then proceeds to extract features by converting text data into their numerical representation using simple bag of words, unigram and bigram. The features which have been converted into their numerical representation are then used to create classification model using the multinomial Naïve Bayes that is quite popular as a shallow machine learning method for text classification that is also could be optimized for various improvements [27], [28]. Finally, we design two main test scenarios as can be seen below, before then continued to evaluate the results and documented all our findings.

- In the first scenario, we isolated the typo correction module to test and evaluate the ability to accurately fix the supposebly misspelled words. We performed manual desk checking by inputting words with various length spanning from 3 lettered words to 12 lettered words.
- In the second scenario, we executed the whole program to build the feedback sentiment classification model. Here, we compared the classification performance between two models, one with the typo correction and another one without typo correction.

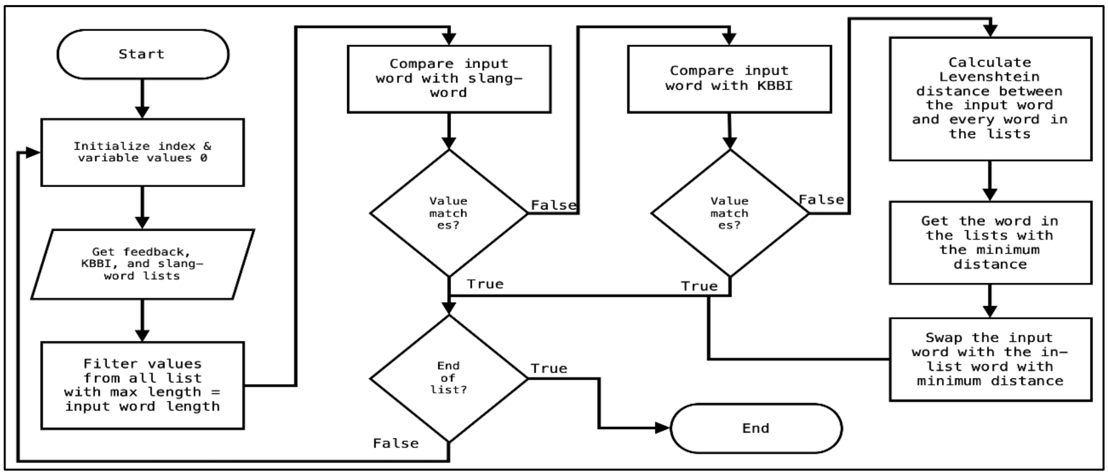

Start
Initialize index & variable values 0
Get feedback, KBBI, and slang-word lists
Filter values from all list with max length = input word length
Compare input word with slang-word
Value matches?
False
True
Compare input word with KBBI
Value matches?
False
True
End of list?
True
False
End
Calculate Levenshtein distance between the input word and every word in the lists
Get the word in the lists with the minimum distance
Swap the input word with the in-list word with minimum distance

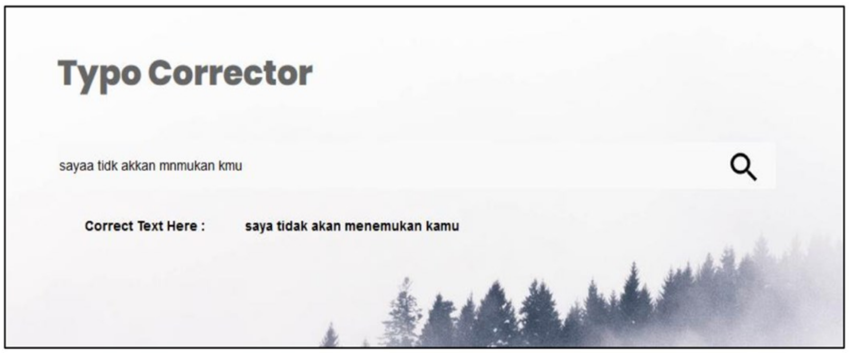
Figure 3. Typo/misspelled word correction flowchart

3. RESULTS AND ANALYSIS

Our experiments are conducted and all modules are built using Python 3 and Jupyter Notebook. We also used several common libraries for working with the data, such as Sastrawi, NLTK, Pandas, and Scikit-Learn. All the program's modules are developed using a computer running on Windows 10 with 8GB RAM, using Intel® Core TM i5-8250U CPU@1.60GHz. This section presents the results of our implementation and findings, including the experiment results which depict the performance of Levenshtein distance algorithm in correcting misspelled words in the dataset used in our research. Moreover, this section also shows the result of the performance comparison between two classification models, one without the typo correction module after data cleansing and the other one implemented the typo correction module. Both classifiers are trained using the same dataset and built using Naïve Bayes classifier and the bag of words feature extraction method.

3.1. Typo correction demo web interface

After the Levenshtein distance algorithm, typo correction sub module, and all the other modules for building the classification model are coded and ready to go, we built a simple local web interface to demo the typo correction using Flask, as shown in Figure 4. In this simple web interface, users can test the same typo correction module, which will also be evaluated more thoroughly in the next sections, by typing a sentence containing misspelled words in Indonesian language and the system will try to correct them and display the result on the screen.



sayaa tidk akkan mnmukan kmu

Correct Text Here : saya tidak akan menemukan kamu

Figure 4. Typo correction demo web interface

3.2. Typo correction performance evaluation

Before implementing the typo correction module in the classification model, we isolated the module and tested the performance of the typo correction module itself which implements the Levenshtein distance algorithm with normalized similarity. As also shown in Table 1, we collected numerous words from the feedback dataset in Indonesian language divided into 10 categories ranging from 3-lettered word to 12-lettered words. Each category is then filled with 21 random words from the dataset, except for the 12-lettered words category which only filled with 11 words. Unsurprisingly, the typo correction module performed poorly with words with small number of letters (14.2% accuracy in 3- and 4-lettered words). The less numbers of letters it has, the more word possibility exist, thus making it more difficult to guess the correct words. However, it performs better as the number of letters increase, as can be seen in Table 1, it achieved a little more than 90% accuracy when dealing with 11- and 12-lettered words. The experiment also revealed that the average time needed for processing one word is 3.84 seconds and the average accuracy of all the words in the experiment is 55.3%.

Table 1. The performance of typo correction module

Length of word	No. of word	Elapsed time (seconds)	Accuracy (%)
3	21	6.7	14.2
4	21	12.8	14.2
5	21	29.3	23.8
6	21	49.3	33.3
7	21	73	42.8
8	21	99	72.4
9	21	116	90.4
10	21	136	80.9
11	21	142	90.4
12	11	75	90.9

3.2. Model comparison

In the last testing scenario, we build two models for feedback sentiment classification, both using Naïve Bayes (α=2.5) and the bag of words feature extraction method. The model uses 482 feedback texts, with a balance 50:50 distribution between text labeled as positive and negative. Within the 482 texts, we used 80% data for training and 20% data for testing the model. The first model only implements common data cleansing processes such as casefolding, removing non-alphanumerical characters, stopwords removal, and stemming. On the other hand, the second model performed the same processes with an additional typo correction module to convert supposebly misspelled words into words existing in the dictionary and slang-word list.

The experiment result showed that the first model without the typo correction module achieved a score of 70.1% of accuracy, while the second model using the typo correction module achieved a score of 78.3% of accuracy. This result shows that albeit the typo correction performed poorly for word with small number of letters, this could generally increase the performance of text classification (8.2% increase), at least in our balanced dataset written in Indonesian language. We also found that even in feedback texts written mainly in Indonesian language, sentences written in combination with foreign languages such as English are often found, which shows that more works to preprocess text with various languages are needed. Furthermore, although more works are also needed to validate these results with various configuration, dataset, and classification techniques, this early result shows the huge potential of employing a more thorough text preprocessing techniques when using supervised machine learning techniques to build text classification models, such as typo correction.

4. CONCLUSION

In this research, we reviewed various literatures related to automated text classification in Indonesian language and how to improve existing metholodogies. This research implemented the Levenshtein distance algorithm with normalized similarity to preprocess supposebly misspelled words in the dataset before the data is used to build a classification model. Furthermore, this paper shows the results of performance evaluation of the typo correction module with Levenshtein distance and comparison of two classification models.

Our experiments show that the Levenshtein distance algorithm works better with longer words in Indonesian language, achieving more than 90% accuracy in 11- and 12-lettered words, however performed poorly in shorter words. The average time needed for processing one word is 3.84 seconds and the average accuracy of all the words in the experiment is 55.3%. Moreover, this paper’s main contribution includes performing a comparison of two text classification models, showing that the model which performs text preprocessing techniques including the typo correction module could achieve an 8.2% increase in accuracy, compared to the model without the typo correction module. These early results show the promising potential of employing various language normalization techniques in data preprocessing especially when dealing with supervised machine learning for texts in Indonesian language, which is the focus of this paper.

Next research steps could include a more thorough validation and evaluation techniques to further reveal the potential of various language normalization and text preprocessing techniques in Indonesian language, combined with techniques to handle foreign languages such as English. Moreover, further research is also needed to evaluate the effectiveness of doing such things when using deep learning techniques and more advanced state-of-the-art word embedding techniques. And finally, sentiment analysis is gaining traction in natural language processing in Indonesian language, future works could include to compare the results in our research to various baseline models and classifiers provided by the research community as well as by the industry.

ACKNOWLEDGEMENTS

This research was supported by the Artificial Intelligence Laboratory in Universitas Multimedia Nusantara. We also thank our colleagues from the Faculty of Engineering and Informatics who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

REFERENCES

[1] S. Kujala, “User involvement: A review of the benefits and challenges,” *Behaviour & Information Technology,* vol. 22, no. 1, pp. 1-16, Jan. 2003, doi: 10.1080/01449290301782.

[2] D. Pagano and B. Bruegge, “User Involvement in Software Evolution Practice: A Case Study,” *35th International Conference on Software Engineering*, San Francisco, May 2013, doi: 10.1109/ICSE.2013.6606645.

[3] B. Regnell and S. Brinkkemper, “Market-Driven Requirements Engineering for Software Products,” *Engineering and Managing Software Requirements*, Springer, pp. 287-308, 2015, doi: 10.1007/3-540-28244-0_13.

[4] E. C. Groen, *et al.,* "The Crowd in Requirements Engineering: The Landscape and Challenges," *IEEE Software,* vol. 34, no. 2, pp. 44-52, Mar. 2017, doi: 10.1109/MS.2017.33.

[5] I. Morales-Ramirez, D. Munante, F. Kifetew, A. Perini, A. Susi, and A. Siena, "Exploiting User Feedback in Tool-Supported Multi-criteria Requirements Prioritization," *25th International Requirements Engineering Conference (RE)*, Sept. 2017, doi: 10.1109/RE.2017.41.

[6] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," *Conference: 21st International Conference on Requirements Engineering*, Jul. 2013, doi: 10.1109/RE.2013.6636712.

[7] L. Derczynski, D. Maynard, G. Rizzo, M. v. Erp, G. Gorrell, R. Troncy, J. Petrak and K. Bontcheva, "Analysis of named entity recognition and linking for tweets," *Information Processing and Management,* vol. 51, no. 2, pp. 32-49, Mar. 2015, doi: 10.1016/j.ipm.2014.10.006.

[8] M. Sanger, U. Leser, and R. Klinger, "Fine-Grained Opinion Mining from Mobile App Reviews with Word Embedding Features," *International Conference on Applications of Natural Language to Information Systems*, Jun. 2017, doi: 10.1007/978-3-319-59569-61.

[9] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble Methods for App Review Classification: An Approach for Software Evolution (N)," *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2015, doi: 10.1109/ASE.2015.88.

[10] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? Classifying user reviews for software maintenance and evolution," *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct. 2015, doi: 10.1109/ICSM.2015.7332474.

[11] E. W. Pamungkas and D. G. P. Putri, "Word Sense Disambiguation for Lexicon-Based Sentiment Analysis," *Proceedings of the 9th International Conference on Machine Learning and Computing*, Feb. 2017, doi: 10.1145/3055635.3056578.

[12] Y. Nurdiansyah, S. Bukhori and R. Hidayat, "Sentiment analysis system for movie review in Bahasa Indonesia using naive bayes classifier method," *Journal of Physics: Conference Series,* Apr. 2018, doi: 10.1088/1742-6596/1008/1/012011.

[13] I. Prasetyaningrum, K. Fathoni and T. T. J. Priyantoro, "Application of recommendation system with AHP method and sentiment analysis," *TELKOMNIKA Telecommunication Computing Electronics and Control,* vol. 18, no. 3, pp. 1343-1353, Jun. 2020, doi: 10.12928/telkomnika.v18i3.14778.

[14] W. Gata and A. Bayhaqy, "Analysis sentiment about islamophobia when Christchurch attack on social media," *TELKOMNIKA Telecommunication Computing Electronics and Control,* vol. 18, no. 4, pp. 1819-1827, Aug. 2020, doi: 10.12928/telkomnika.v18i4.14179.

[15] 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 Computing Electronics and Control,* vol. 18, no. 2, pp. 799-806, Aug. 2020, doi: 10.12928/telkomnika.v18i4.14179.

[16] I. Ferdino and A. Rusli, "Using Naïve Bayes Classifier for Application Feedback Classification and Management in Bahasa Indonesia," *5th International Conference on New Media Studies (CONMEDIA)*, 2019, doi: 10.1109/CONMEDIA46929.2019.8981830.

[17] G. P. Wiratama and A. Rusli, "Sentiment Analysis of Application User Feedback in Bahasa Indonesia Using Multinomial Naïve Bayes," *CONMEDIA*, Oct. 2019, doi: 10.1109/CONMEDIA46929.2019.8981850.

[18] A. Rusli, J. C. Young and N. M. S. Iswari, "Identifying Fake News in Indonesian via Supervised Binary Text Classification," *IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Jul. 2020, doi: 10.1109/IAICT50021.2020.9172020.

[19] A. Rusli, A. Suryadibrata, S. B. Nusantara and J. C. Young, "A Comparison of Traditional Machine Learning Approaches for Supervised Feedback Classification in Bahasa Indonesia," *International Journal of New Media Technologies,* vol. 7, no. 1, pp. 28-32, 2020, doi: 10.31937/ijnmt.v1i1.1485.

[20] L. H. Nguyen, A. Salopek, L. Zhao and F. Jin, "A natural language normalization approach to enhance social media text reasoning," *IEEE International Conference on Big Data (Big Data)*, Dec. 2017, doi: 10.1109/BigData.2017.8258148.

[21] M. K. Dalal and M. A. Zaveri, "Automatic Text Classification: A Technical Review," *International Journal of Computer Applications,* vol. 28, no. 2, pp. 37-40, Aug. 2011, doi: 10.5120/3358-4633.

[22] W. Zhang and F. Gao, "An improvement to naive bayes for text classification," *Procedia Engineering,* vol. 15, pp. 2016-2164, 2011, doi: 10.1016/j.proeng.2011.08.404.

[23] S. Zhang, Y. Hu and G. Bian, "Research on string similarity algorithm based on Levenshtein Distance," *IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Mar. 2017, doi: 10.1109/IAEAC.2017.8054419.

[24] J. Schepens, A. Dijkstra and F. Grootjen, "Distributions of cognates in Europe based on the Levenshtein Distance," *Radboud University Nijmegen*, Jan. 2008, doi: 10.1017/S1366728910000623.

[25] Kateglo.com, "Kateglo-Kamus, tesaurus, dan glosarium Bahasa Indonesia," 2009. [Online]. Available: https://kateglo.com

[26] A. Maulana, "Github," Jul. 2018. [Online]. Available: https://github.com/anggamaulana/svm-firefly-optimization/blob/master/slangwords.txt

[27] B. Tang, S. Kay and H. He, "Toward Optimal Feature Selection in Naive Bayes for Text Categorization," *IEEE Transactions on Knowledge and Data Engineering,* vol. 28, no. 9, pp. 2508-2521, Feb. 2016, doi: 10.1109/TKDE.2016.2563436.

[28] B. Wagh, J. V. Shinde and P. A. Kale, "A Twitter Sentiment Analysis Using NLTK and Machine Learning Techniques," *International Journal of Emerging Research in Management & Technology,* vol. 6, no. 12, pp. 37-44, Jun. 2018, doi: 10.23956/ijermt.v6i12.32.

BIOGRAPHIES OF AUTHORS

Reza Setiabudi is an undergraduate student from the Department of Informatics in Universitas Multimedia Nusantara, Indonesia. He has a background in research in applied machine learning for natural language processing in Indonesian language. His interest includes the development of automation testing using various tools in Python, such as Magic Mock, Pytest, and Django Factory.

Ni Made Satvika Iswari received her bachelor's and master's degree from Institut Teknologi Bandung (ITB), Indonesia in 2013. She is a lecturer in Department of Informatics, Universitas Multimedia Nusantara, Indonesia. She is currently pursuing the Ph.D. degree in computer science from Universitas Indonesia. Her research interests include e-business platform and software engineering.

Andre Rusli received his master's degree in Information Environment from Tokyo Denki University, Japan, in 2017. He is currently pursuing a doctoral degree in the Graduate School of Advanced Science and Technology in Tokyo Denki University, Japan, while also serving as a lecturer in Universitas Multimedia Nusantara. His research interests include requirements engineering in software application development, natural language processing, and human computer interaction.