ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse217952020.pdf
https://doi.org/10.30534/ijatcse/2020/217952020

Evaluating RNN Architectures for Handling Imbalanced Dataset in Multi-Class Text Classification in Bahasa Indonesia

Christianto[1], Julio Christian Young[2], Andre Rusli[3]
[1]Universitas Multimedia Nusantara, Indonesia, christianto1@student.umn.ac.id
[2]Universitas Multimedia Nusantara, Indonesia, julio.christian@umn.ac.id
[3]Universitas Multimedia Nusantara, Indonesia, andrerusli19@gmail.com

ABSTRACT

COVID-19 pandemic makes students can ony continue their education through the E-Learning system. In order to fulfill the goal of E-Learning, learning center departments of educational institutes need to know what the user needs and the only way to communicate with them is through feedback. However, it is a hard and time-consuming task to extract value from a large amount of feedback. This paper aims to implement and evaluate several RNN architectures' performances including Simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), to be able to classify feedback text to its categories via a multiclass classification approach. Furthermore, this paper uses FastText in comparison with Keras Embedding Layer to extract features along with the use of Random Oversampling and SMOTE in order to deal with imbalanced dataset problem. Based on the result, our final model could achieve a macro-averaged F1-Score of 64.35% using LSTM architectures. Furthermore, our paper shows that FastText has a poor performance in every RNN architectures and Random Oversampling has a better performance than SMOTE in handling imbalanced dataset problem.

Key words : Bahasa Indonesia, Feedbacks, Imbalanced Dataset, Multi-class Text Classification, Recurrent Neural Network

1. INTRODUCTION

Education is the most important tool in increasing the success rate of a person in life. It helps a person to have a good career in our life. The highly educated we are, the better chance we get [1]. However, during this COVID-19 pandemic, students can only continue their education through E-Learning via the school's online system. According to [2], E-Learning is anything that can be delivered, enabled, or mediated by electronic technology for the explicit goal of learning. So basically, E-Learning is an educational system that has its own requirement in order to fulfill the goal of learning via electronic technology.

Every system requirement is based on its user to satisfy their needs. That is why feedback is the most effective way for a system to improve its user convenience seeing that feedback is the only way to communicate with them. With feedback, users can state any issues that they found in the system. Therefore, it helps the learning center departments who are usually in charge of the E-Learning system in educational institutes, to understand user requirements such as which features should be maintained and which features need to be improved in the system.

However, it is hard to extract the user requirement from a large amount of feedback. According to IBM, text is one type of unstructured data due to its cluttered nature. Therefore, analyzing and understanding the value from text data is a hard and time-consuming task which makes most companies fail to extract value from it [3]. In addition, it is hard to find minor categories feedback such as bug report in thousands of feedbacks, considering that not many people experience bug at the system meanwhile bug need to be dealt quickly. Therefore, it is proposed the application of machine learning methods to classify the feedback categories from the large data sources which is called text classification [4].

Text classification is the process of specifying text documents to one or more proper category based on their features by building a model through a training data [5]. These days, text classification is a necessity due to a very large amount of digital text documents that need to be dealt daily [6]. However, the messy nature of text, a large number of attributes, and imbalanced dataset are problems for building a proper model for text classification. These problems can be handled through the feature selection phase such as

punctuation removal, stopwords removal, stemming, etc. [5].

Most of the text classification systems used are designed to handle English languages; our research mainly focuses on feedbacks written in Bahasa Indonesia. Classic algorithms such as Naïve Bayes and Support Vector Machine (SVM) are often used to classify texts in Bahasa Indonesia for various purposes [7, 8, 9]. Furthermore, SVM and Naïve Bayes have always been the most popular classification algorithm for many years because of its effectiveness and performance in the classification task [5, 10]. However, recent studies regarding Recurrent Neural Network (RNN) shows that RNN achieves better results than SVM and Naïve Bayes in term of text classification [11, 12, 13]. This paper aims to present our preliminary results of implementing and evaluating several RNN architectures' performances including Simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), to be able to classify feedback text to its categories via a multiclass classification approach. Furthermore, this paper uses FastText in comparison with Keras Embedding Layer to extract features along with the use of Random Oversampling and SMOTE in order to deal with imbalanced dataset problem.

2. DATASET

We conducted our experiment using the dataset provided from learning department center of Universitas Multimedia Nusantara which contained 6,558 feedbacks written in Bahasa Indonesia, collected from many learners and instructors in the university. Each feedback is labelled with Accessibility, Bug Report, Feature Request, Helpdesk, User Interface, or Other label. Table 1 below shows the number of feedbacks for each label, along with snippets of examples for each category. As shown on the table, there are 1,384 feedbacks classified as feedback about accessibility problem, 84 feedbacks classified as feedback about reporting bug, 685 feedbacks classified as feedback about feature request, 244 feedbacks classified as feedback about the learning center department, 370 feedbacks classified as feedback about the system's user interface, and other than that, classified as other feedback.

Table 1: Data Distribution

Label	Number of feedbacks	Snippet of a sample feedback
Accessibility	1384	"… bisa lebih diperhatikan lagi tingkat kestabilan server agar lebih lancar saat diakses dan tidak cepat overload"
Bug Report	84	"… beberapa assignment tugas tidak muncul di timeline"
Feature Request	685	"Adakan e-learning dalam bentuk aplikasi mobile yang dilengkapi fitur notifikasi …"
Helpdesk	244	"… perlu adanya pemberitahuan awal melalui web e-learning apabila akan ada perbaikan/pengembangan sistem …"
User Interface	370	"… temukan ui ux yang dapat mudah dipahami mahasiswa"
Other	3789	"sudah bagus ditingkatkan lagi"

Based on the dataset, we use Python 3 and Jupyter Notebook provided by Google Collaboratory [23] to conduct our research.

3. METHODOLOGY

3.1 Word Embedding

Machine learning algorithms can automatically learn, extract, and analyze features by taking numeric feature vectors as its input [14]. Therefore, when dealing with text data classification, the words in the text are usually vectorized into a numerical format [15]. For this purpose, several methods exist and one of them is word embedding. This paper uses FastText and Keras Embedding Layer as word embedding methods.

A. FastText

FastText is a library deriving word embeddingsdeveloped by the Facebook Research team,that implements a very efficient text classifier[16]. FastText generates word embeddings similar toWord2Vec, but FastText has the upper hand bypresenting char n-grams that facilitates the learning of uncommon words [17].

B. Keras Embedding Layer

Keras Embedding Layer is an embedding layer in neural networks that turns positive integers into dense vector of fixed size [24]. It needs the input data to be integer encoded, thus every word is represented by a unique integer. Thereafter, it initializes random weights for every word and learns the embedding for all of the words within the training dataset [18].

3.2 Resampling Methods

Resampling methods aim to vary the dataset in order to reduce discrepancy among the sizes of the classes, so we use one of resampling methods called over-sampling in order to deal with imbalanced dataset. Over-sampling is a method to generate data in order to increase the size of minority class. This paper uses SMOTE and Random oversampling as resampling methods.

A. SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) [19] is an oversampling method based on creating synthetic instances for minority classes. The minority class is over-sampled by fetching each minority class sample and forming artificial instances along with the line segments by joining any or all of the *k* minority class nearest neighbors. The *k* nearest neighbors are randomly chosen depending on the amount of over-sampling needed. The synthetic instances cause the classifier to create larger and less specific decision region that makes general regions learned for the minority class rather than being subsumed by majority class.

B. Random Oversampling

Random oversampling is a simple approach that takes samples at random from minority class and duplicate the samples, so it reaches a size comparable with the majority class [17].

3.3 Classification Algorithm

For classification algorithm, we use Recurrent Neural Network (RNN) including three different RNN architectures; that is, Simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

A. Simple RNN

RNN makes use of sequential information, therefore the output relies on the previous computation. The advantage of RNN is having a memory that captures information in arbitrary long sequence [12]. RNN model analyzes text word by word, where the semantic of the previous texts is preserved in a fixed-sized hidden layer [20]. However, RNN is a biased model, because recent words are more significant than the previous one. Therefore, the key components could appear anywhere across the document and might reduce the performance when used to capture the semantic of whole documents. That is why GRU and LSTM model are introduced to overcome RNN difficulties [12].

B. LSTM

LSTM is more complicated than Simple RNN,it learns to control the flow of information in order to prevent the vanishing gradient problem and let the recurrent layer to capture long-term dependencies more easily. RNN has problems of gradient vanishing or explosion and LSTM overcome these issues with a new structure called memory cell. The memory cell consists four main components; that is, input, output, forget gates, and a candidate memory cell [12].

C. GRU

GRU approach relies on dynamical analysis that suffers from gradients explosion because of their non-linear dynamic. It was designed to properly update or reset its memory contents and is a lightly simplify variation of LSTM [21]. It is a combination of the input and forget gates into one "update gate" and has an extra gate called "reset gate". GRU model is simpler and has fewer parameters when compared to traditional LSTM models [22].

3.4 Text Preprocessing

First, the dataset is provided in Microsoft Excel Format (.xlsx) imported as a Python data frame object. Each feedback is preprocessed by case-folding it into lowercase characters and then removing numbers and punctuation. Furthermore, we remove all the stop-words and all affixes with stemming. Figure 1 shows the step-by-step activities which are performed, from importing the dataset until the train-test data split, which then is followed by the feature extraction and model selection phase.

Following the text-preprocessing step, the dataset is then split into training, testing, and validation set with an 70:15:15 ratio. In order to deal with imbalanced dataset, we use oversampling techniques on training set. Therefore, the classification algorithm will have a proportionate ratio of observations for each class. We use tokenizer first that convert text data into numeric value in order to use oversampling techniques. We compared the result of oversampling training set using SMOTE and Random Oversampling.

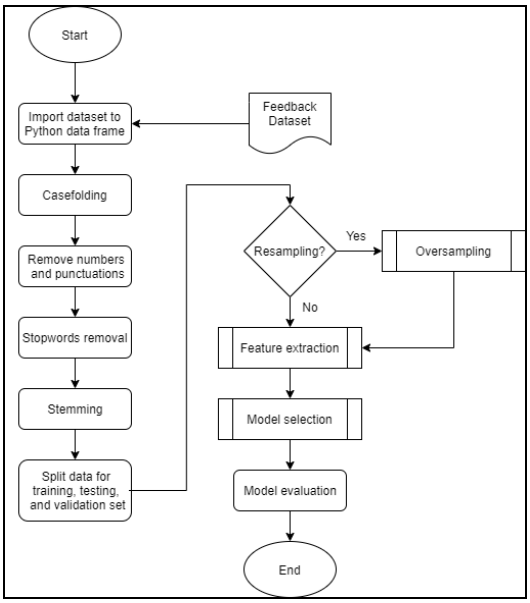

Figure 1: Feedback Classification Steps

Following the resampling process, the features are extracted to numerical data in order to build the classification model. We compared the result of extracting features using FastText and Keras Embedding Layer. Both of them convert a collection of raw documents into a matrix of weights.

Finally, after the features are extracted using the previous approaches, the Recurrent Neural Network classifier is

prepared. Several hyperparameters configurations were used and compared such as epoch, batch size, learning rate, hidden layers, and RNN architecture, in order to achieve the best result in classifying feedbacks in testing set. We evaluate the models with macro-averaging because of the imbalanced dataset problem and we want to weight our metric towards the minority class.

4. RESULT AND DISCUSSION

Our results are generally divided firstly into two categories, with and without oversampling method. Secondly, using Keras Embedding Layer or FastText as feature extraction method. For each scenario, we compare the macro average score between LSTM, GRU, and Simple RNN architecture in classifying feedback. In order to purely compare the RNN architectures, we use the same hyperparameter for each architecture that we configure first; that is, learning rate = 0.01, epoch = 20, hidden node = 128 on RNN architecture, a bidirectional layer for RNN architecture, a dense layer with hidden node = 128 and relu as activation function, batch size = 32 for oversampling scenario, and batch size = 16 for no oversampling scenario.

4.1 Simple RNN

In the first experiment, we use Simple RNN architecture as our classifier. Table 2 displays the result of Simple RNN classification with different configurations of feature extraction and oversampling method.

Table 2: Simple RNN Classification Results

Classifier	Feature Extraction	Oversampling	Macro-F1 Score
Simple RNN	Keras Embedding Layer	-	54.59%
		ROS	56.39%
		SMOTE	50.25%
	FastText	-	34.30%
		ROS	29.22%
		SMOTE	28.82%

The result shows that Simple RNN architecture has the best performance using Keras Embedding Layer and Random Oversampling with macro-averaged F1-score of 56.39%. Surprisingly FastText which popular as feature extraction method, has poor performance in all oversampling scenario. On the other hand, the result shows that Random Oversampling has better performance than SMOTE as oversampling method.

4.2 LSTM RNN

Afterward, we use LSTM RNN architecture as our classifier. Table 3 displays the result of LSTM classification with different configurations of feature extraction and oversampling method.

Table 3: LSTM RNN Classification Results

Classifier	Feature Extraction	Oversampling	Macro-F1 Score
LSTM	Keras Embedding Layer	-	64.35%
		ROS	63.25%
		SMOTE	52.15%
	FastText	-	38.63%
		ROS	36.69%
		SMOTE	35.74%

The result shows that LSTM architecture has the best performance using Keras Embedding Layer without any oversampling method, with macro-averaged F1-score of 64.35%. However, FastText shows a poor performance once again with LSTM in all oversampling scenario. Unlike Simple RNN, Random Oversampling slightly deteriorate the performance of LSTM classification.

4.3 GRU RNN

At last, we use GRU RNN architecture as our classifier. Table 4 displays the result of GRU classification with different configurations of feature extraction and oversampling method.

Table 4: GRU RNN Classification Results

Classifier	Feature Extraction	Oversampling	Macro-F1 Score
GRU	Keras Embedding Layer	-	61.34%
		ROS	63.31%
		SMOTE	50.84%
	FastText	-	40.84%
		ROS	32.66%
		SMOTE	31.55%

The result shows that GRU architecture has the best performance using Keras Embedding Layer and Random Oversampling with macro-averaged F1-score of 63.31%. Like the other architectures, FastText shows a poor performance with GRU in all oversampling scenario.

Based on the results, our final model could achieve a macro-averaged F1-score of 64.35%. This is achieved by LSTM RNN classifier using Keras Embedding Layer without

any oversampling method. However, we also found that FastText has a poor performance in every RNN architectures. This either indicates that FastText is not compatible with RNN architecture or FastText cannot handle imbalanced dataset problem. Moreover, we also found that Random Oversampling has better performance than SMOTE in handling imbalanced dataset problem seeing that Random Oversampling always have a higher value of macro-averaged F1-score in every RNN architectures.

5. CONCLUSION

COVID-19 pandemic makes students can only continue their education through the E-Learning system. In order to fulfill the goal of E-Learning, learning center departments of educational institutes need to know what the user needs and the only way to communicate with them is through feedback. However, it is a hard and time-consuming task to extract value from a large amount of feedback. Our paper's contribution is to implement and evaluate several RNN architectures' performances in classifying feedback text to its categories along with various configurations of feature extraction and oversampling methods.

Based on the classification report, our final model could achieve a macro-averaged F1-score of 64.35%. This is achieved by LSTM RNN classifier using Keras Embedding Layer without any oversampling method. The hyperparameter configuration that we used on the classifier are a bidirectional layer on RNN architecture, hidden node = 128 on RNN architecture, a dense layer with hidden node = 128 and relu as activation function, batch size = 16, epoch = 20, and learning rate = 0.01. However, based on other experimentations, we also found that FastText has a poor performance in every RNN architectures. This either indicates that FastText is not compatible with RNN architecture or FastText cannot handle imbalanced dataset problem. Moreover, we also found that Random Oversampling has better performance than SMOTE in handling imbalanced dataset problem. However, Random Oversampling slightly deteriorate the performance of LSTM classification.

In this research, ourfocus is on the RNN architecture's configuration. Therefore, future works could implement hyperparameter configuration on the feature extraction and oversampling method because we tried to configure embedding dimension parameter a little and it has an effect on the model's performance. In addition, future works could implement various feature engineering methods such as bag of words and traditional machine learning classification methods because previous work shows that traditional machine learning could have a better performance than RNN architectures [25].

ACKNOWLEDGEMENT

This research was supported by the Artificial Intelligence Laboratory in Universitas Multimedia Nusantara. We also thank Learning Center Department of Universitas Multimedia Nusantara for providing the dataset that we used in this research.

REFERENCES

1. Al-Shuaibi, Abdulghani.**The Importance of Education**. 2014.
2. K. Cheng.**A Research Study on Students' Level of Acceptance in Applying E-Learning for Business Courses – A Case Study on a Technical College in Taiwan**. Journal of American Academy of Business, Vol. 8. No. 2. pp. 265-270, 2006.
3. Schneider, Christie. **The Biggest Data Challenges That You Might Not Even Know You Have**. Watson Blog, May 24, 2019. https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/.
4. Nurhuda F, Sihwi S.W, and Doewes, A. **Analisis Sentimen Masyarakat terhadap Calon Presiden Indonesia 2014 berdasarkan Opini dari Twitter Menggunakan Metode Naive Bayes Classifier**. Jurnal Itsmart, 2(2), pp. 35–42, 2013.
5. Mowafy M, Rezk A,and El-Bakry H.M. **An Efficient Classification Model for Unstructured Text Document**. American Journal of Computer Science and Information Technology,Vol.6 No.1: 16, 2018.https://doi.org/10.21767/2349-3917.100016.
6. Ikonomakis E, Kotsiantis S, and Tampakas V. **Text Classification Using Machine Learning Techniques**. WSEAS transactions on computers,Vol.4, pp. 966-974, 2005.
7. R. Wongso, F. A. Luwinda, B. C. Trisnajaya, O. Rusli and Rudy.**News Article Text Classification in Indonesian Language**. 2nd International Conference on Computer Science and Computational Intelligence, Bali, 2017.
8. G. P. Wiratama and A. Rusli. **Sentiment Analysis of Application User Feedback in Bahasa Indonesia Using Multinomial Naïve Bayes**. 5th International Conference on New Media Studies (CONMEDIA), Bali, 2019.
9. 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), Bali, 2019.
10. Goudjil M, Koudil M, Bedda M, and Ghoggali N. **A Novel Active Learning Method Using SVM for Text Classification**. *International Journal of Automation and Computing*, *15*(3), pp. 290–298, 2016. https://doi.org/10.1007/s11633-015-0912-z.

11. Y. Hu, Y. Li, T. Yang and Q. Pan. **Short Text Classification with A Convolutional Neural Networks Based Method**. 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore,pp. 1432-1435, 2018.https://doi.org/10.1109/ICARCV.2018.8581332.
12. A. Hassan and A. Mahmood. **Efficient Deep Learning Model for Text Classification Based on Recurrent and Convolutional Layers**. 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun,pp. 1108-1113, 2017.https://doi.org/10.1109/ICMLA.2017.00009.
13. Zhou X, Wan X, and Xiao J.**Attention-based LSTM Network for Cross-Lingual Sentiment Classification**. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016. https://doi.org/10.18653/v1/d16-1024.
14. Syamala M andNalini N. J. **A Deep Analysis on Aspect based Sentiment Text Classification Approaches**. Int. J. Adv. Trends Comput. Sci. Eng.,vol. 4, no. 2, pp. 15–21, 2019. https://doi.org/10.30534/ijatcse/2019/01852019.
15. Sueno H. T, Gerardo B. D, and Medina R. P. **Multi-class Document Classification using Support Vector Machine (SVM) Based on Improved Naïve Bayes Vectorization Technique**. Int. J. Adv. Trends Comput. Sci.Eng.,vol. 9, no. 3, pp. 3937–3944, 2020. https://doi.org/10.30534/ijatcse/2020/216932020.
16. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, **Enriching word vectors with subword information**.Transactions of the Association for Computational Linguistics, Vol. 5, 2017.
17. Padurariu C and Breaban M. E.**Dealing with Data Imbalance in Text Classification**. Procedia Computer Science, 159, pp. 736-745, 2019.
18. Brownlee, J. **How to Use Word Embedding Layers for Deep Learning with Keras**, Machine Learning Mastery, 2019. https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/.
19. N. Chawla, K. Bowyer, L. O. Hall, and W. Philip Kegelmeyer.**Smote: Synthetic minority over-sampling technique**.Vol. 16, 01, 2002.
20. Elman, J.L. **Finding structure in time**. Cognitive science,14(2), pp. 179-211, 1990.
21. Zulqarnain M, Ishak S.A, Ghazali R, andNawi N.M. An **Improved Deep Learning Approach based on Variant Two-State Gated Recurrent Unit and Word Embeddings for Sentiment Classification**. Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 1, pp. 594–603, 2020.
22. Zulqarnain M, Ghazali R, HassimY.M.M, and Rehan, M. **Text classification based on gated recurrent unit combines with support vector machine**. InternationalJournal of Electrical and Computer Engineering (IJECE), 10(4), 3734, 2020. https://doi.org/10.11591/ijece.v10i4.pp3734-3742.
23. Google, **Google Colaboratory**, Google, 2020. https://colab.research.google.com/.
24. Chollet, F.**keras**, GitHub, 2015. https://github.com/fchollet/keras.
25. Al-Smadi M, Qawasmeh O, Al-Ayyoub M, Jararweh Y, and Gupta B. **Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews**. Journal of Computational Science, 27, pp. 386-393, 2018. https://doi.org/10.1016/j.jocs.2017.11.006.