ISSN 2085-4552

FastText Word Embedding and Random Forest Classifier for User Feedback Sentiment Classification in Bahasa Indonesia

Yehezkiel Gunawan[1], Julio Christian Young[2], Andre Rusli[3]
Department of Informatics, Multimedia Nusantara University, Tangerang, Indonesia
[1]yehezkiel.gunawan@student.umn.ac.id, [2]julio.christian@umn.ac.id, [3]andre.rusli@lecturer.umn.ac.id

Accepted 14 June 2021
Approved 18 June 2021

***Abstract*— User feedback nowadays become a platform for software developer to identify and understand user requirements, preferences, and user's complaints. It is important for the developer to identify the problem that exist in user feedback. According to software growth, user amount also growth. Read and classify one by one manually are wasting time and energy. As the solution for the problem, sentiment analysis system using Random Forest Classifier which use word embedding as the feature extraction is made to help to classify which feedback is positive, neutral, or negative. Random Forest Algorithm is chosen because it gives the best performance, even its need the larger resources. Furthermore, with word embedding, the words which has semantic or syntactic similarities will be detected. Word embedding does not need stemming and stop word removal, so the context of the sentences keep remains. This research is made to implement word embedding to classify sentiment of user feedbacks using Random Forest Classifier. 70.27% accuracy, 80% precision, 54 recall and 54% F1 score is reached when BYU dataset (200 dimension) as embedding dataset with the train and test ratio 80:20.**

Index Terms*— Bahasa Indonesia, Random Forest, Word Embedding, NLP, user feedback*

I. INTRODUCTION

In software engineering, there is a term called 'requirements engineering'. It means a process which the requirements for a software are assembled, documented, and managed as long as the software engineering process [1]. Interpreting and understanding the purpose, stakeholder's requirements and trust are the main focus of requirement engineering [1].

It was shown in [2] that nowadays users or clients are involved in the software engineering process, so the developers will know the needs, preferences, and problems which users experienced. Software developers must understand the issues or problem that appear from the app which they developed, like bugs, unwanted feature, and adding the new feature which accurate and on time in the future [3]. Users who experience some issues when they use an app can send some feedbacks that can reflect their experience when they use it, then the feedbacks can be considered by the developers to improve the app quality [3].

When the scope of the application are become greater, it is much challenging to identify user feedbacks issues [4]. Modern problems require modern solutions, so a recommendation system or sentiment classification needs to be made regarding to the growth of user feedbacks [4]. There are more than a hundred or even thousand user feedbacks have been sent in one day. Checking users' comment one by one can be exhausting and wasting time. That is the reason why automated sentiment analysis is needed to analyze and generalize the user's feedback with the sentiment analysis technique.

A method called CRISTAL is introduced by Palomba *et al* [4] which can detect the impact caused by the informative user comment on the changes or updates of the app's source code. The research is done to determine how impactful the app review on the software development process [4]. Its result stated that 49% of developers will consider the informative user comments or feedback for their next updates. It also considered very impactfully on the app success because the increase of the rating and positive feedback from the users directly compared with the fulfillment of the requirements based on the user review. This result can strengthen the reason why we need a system to recommend or classify the user feedbacks to help the developer on developing their app [4].

Sentiment analysis is the process that learn people's opinion, sentiment, emotion, rating, and gesture on an entity [5]. There are a lot of activities that are related to the sentiment analysis process and even harder to separate it because there are a lot of aspects, and one of them is sentiment classification. Sentiment classification is based on the idea that text can be the expression of a person's opinion on an entity and trying to predict what kind of sentiment that

can be resulted [6]. Machine learning in general can be used to classify the sentiment and give the good accuracy.

As in [7], sentiment analysis is an automated process to mine and classify opinion, view, emotion, and sentiment from the text dataset which are not structured for machine language and computer programming. In sentiment classification, text can be classified to several labels, for example positive or negative.

Nowadays, people express their opinion with their language which tends to be ambiguous and complicated words [8]. Commonly, there are related words one each other and it often seems similar. To help to solve the problem, there is a method called word embedding. It is a kind of word representation that make the words which have similarity can be understood by the machine learning algorithm [9]. Technically, the input words will be mapped into number vector using neural network, probability model, or the dimension reduction on the word co-occurrence matrix.

It is stated in [10] that word embedding is considered can learn the word vector with high quality from a big dataset. Instead of that, the existing vocabulary that has been made from the pre-trained model of the word embedding also considered detecting the word similarity both semantically or syntactically. It can help the machine to recognize the similarity which exists in the dataset.

Currently, two big platforms that provide the apps choice and review, are Google Playstore and Apple Appstore. The user feedback dataset from these platforms is considered because when a user wants to give some feedback to an app, he or she must have an account so it will allow the user to give some review. For example, when a person wants to give some feedback to an app in Google Playstore or an Apple Appstore, so that person must have a Google or Apple account to allow him or her to write the feedback on their smartphone. Besides that, some developers can set the app which they made to allow the user to give some review within a certain period, so that can be confirmed if the user who gives the feedback is the user of the app.

This research will use *word embedding* as the feature extraction and Random Forest Classifier to classify the user feedback's sentiment. This research aims to study, analyze, and implement the Random Forest Classifier to analyzing the sentiment of application user feedbacks in Bahasa Indonesia. The paper is organized as follows. In the following section, we review some related works of ours. Then we present a brief overview of some research in sentiment analysis.

II. Related Work

A comparative study has been done by [8] on few machine learning algorithms used for classification. Those are Naïve Bayes, Max Extropy, Boosted Trees, and Random Forest. The result of the research is Random Forest has the best performance with the high simplicity even it requires more resources.

Following similar trends, several works of literature can also be found working on sentiment analysis of documents. However, most of them focus on analyzing tweets from Twitter. Like similar observation done by Vora, Khara, and Kelkar [9], they used different word embedding methods as the feature extraction to classify the sentiment of English tweet. For the classification algorithm, they used Random Forest Classifier. The result shows that when they used FastText with 300 dimensions as the feature extraction, the accuracy reached 91%.

Based on those previous works on sentiment analysis, natural language processing, and the requirements engineering activity in the software engineering field, this research focuses on adapting [9] research but with different dataset. Our research aims to implement FastText and Random Forest Classifier to classify the sentiment of application user feedback in Bahasa Indonesia and measure the performance.

III. Research Methods

A. Requirements Engineering

Requirement engineering is the process of gathering, analyze, documenting, and managing the requirements needed for software development [1]. It always related to determine and understand the purpose, requirements, and even what the stakeholder trust [1].

In requirement engineering, some things need to be done, those are feasibility study, finding the requirements (gathering information and analyze), convert the requirements into the standard or specification, and ensure that the requirements are based on the user's need (validation) [11]. In reality, this process is iterative and interleaved [11].

According to [11], in the requirement gathering process, the client is involved to determine the scope of the app, what services will exist in the system and the operational limits of the system. It may be involved the user, manager, engineer, and the people who will maintain the system. In all system, the requirements can be changed, the people who involved, developed a better understanding of what are they want in the app when it's released, like the changes on the hardware, software, or the system environment [11]. The understanding and control process of changes on the system requirement is called

requirement management [11]. Besides that, the feedback from the user can cause changes to the requirements [2].

The software development activity [11] is not stopped when the software has been released. But it continues throughout its lifetime. There are five steps of the software lifetime, those are the initial development, the software engineer build the first version of the app or system, then evolve, the ability and functionality of the system are expanded to fulfill the user preferences, then servicing, the system is fixed from the bug, issues, and update the functionality, then phaseout, the system owner decided to stop the servicing process and make income from the system for a long time, and the last step is closedown, the owner takes down the system from the market and direct the users to the new system.

B. Natural Language Processing

Natural language processing (NLP) is a computer science field dealing with human language processing in either text or speech [12]. In this research, preprocessing of user feedback includes the punctuation, remove special character, lowering case, and tokenization. All that preprocessing method is done by the help of library string and re.

C. Word Embedding

It is shown in [13] that the computer can learn the character input with feature extraction. Every kind of feature is taken from the dataset, then the machine will learn it. In this research, the feature which will be extracted is the word similarity.

There is a problem that has to be faced. The computer only can read the numbers. If the received objects are words, so it needs to be converted into the numeric vector which represents each word. Because of that, word embedding can be used as a feature extraction to learn the similarity between words. It uses a neural network model to learn the words [13].

Word embedding represent the words into vectors. For example, there is the sentence Word Embedding are Word Converted into numbers. First, it will make a dictionary to contain it, and the dictionary is [Word, Embeddings, are, Converted, into, number]. With one-hot encoding, it will represent the vector where 1 represent the position of a word. The vector representation of word numbers from that example is [0,0,0,0,1].

This method learns the vector representation of the constant vocabulary which exists in the corpus or dataset. It also uses neural network model for the task like document classification or with unsupervised learning using document statistics [13]. In general, three common models are often used to do word embedding, those are Latent Semantic Analysis (LSA), Word2Vec, and GloVe.

D. FastText

This research will use FastText as the word embedding method. FastText is the library from Facebook to do the word embedding method [14]. FastText is a newest version from Word2Vec which Google made. Actually, both of them can be used to determine the word similarity.

According to the Indonesian Dictionary, semantic means the language structure related to the meaning of an expression or the structure of the meaning of a talk or text. Meanwhile, the syntactic comes from the 'syntax' word adopted in English which is the structure or writing. In other words, if there is a word similarity syntactically, it means there is a similarity in the writing structure.

The input words will be represented into the vector and placed in such a way so the words which have the similar meaning will appear close by, meanwhile, the opposite will appear far from the vector. The main difference between FastText and Word2Vec is FastText can process the input words which not exist in the vocabulary or out of vocabulary words.

Like Word2Vec, there is two architecture on FastText, those are Continuous Bag of Words (CBOW) and Skip-Gram. CBOW predicts the current word (as a target) from the context (as an input) around it. Meanwhile, the Skip-Gram uses the current word (as an input) to predict the context (as a target). The visualization of CBOW and Skip-Gram can be seen in Fig.1

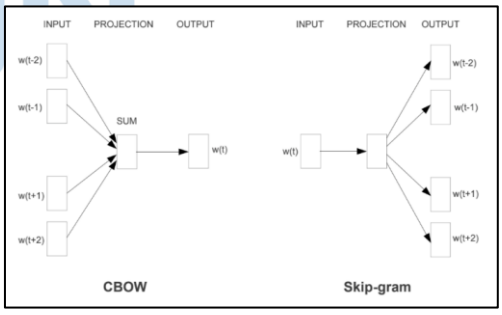

Fig. 1. The concept of CBOW and Skip-Gram [10]

In the Fig.2, there is an example which given the input words "the best revenge is massive success" and there is a forward-backward training with the CBOW architecture. Assume that w(t-2) = "the", w(t-1) = "best", w(t+1) = "is", w(t+2) = "massive" as input and w(t) = "revenge" as target.

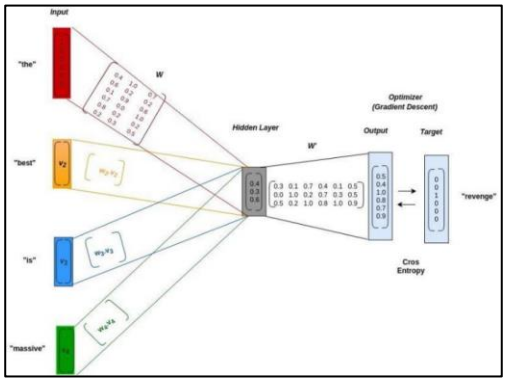

Fig. 2. Ilustration of forward-backward training CBOW [10]

[14] said it requires a special dataset to get the expected result. [14] makes a term for the dataset which used for the training process using FastText as an embedding dataset. The embedding dataset trained using FastText will produce a vocabulary that consists of the vocabs that can be used to detect the word similarity. The result of the training of the process can be called a pre-trained model.

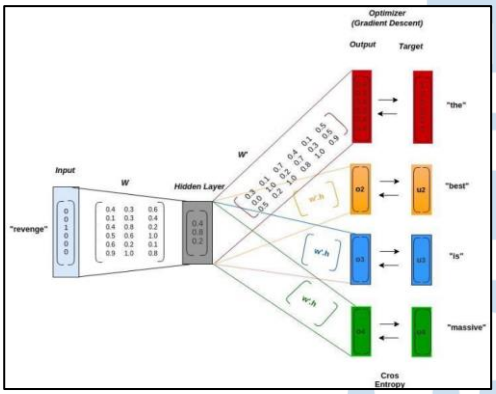

Fig. 3. Ilustration of forward-backward training Skip-Gram[10]

E. Decision Tree

Decision Tree is a machine learning method that learns and take decision with the functional target that has discrete values [15]. This technique can be represented with a group of if-then rules so it will be understandable. Each tree consist of the leaf and branch. Each leaf reflect the group attribute which will be classified and each branch represent the values which taken by the leaf.

There are three parts of decision tree like the Fig.4 [16]. First is the root node which is the first node and there is no input branch in this node. The second is internal node which has branch, and just has one input and minimum two output. The last is leaf node that is the node which has just one input and no output.

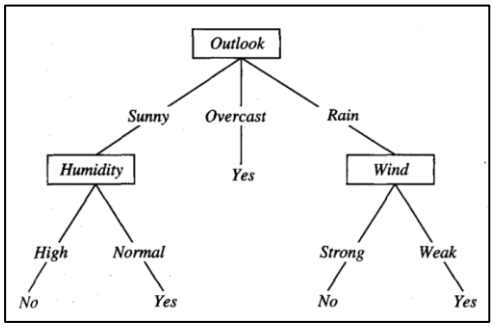

Fig. 4. Decision Tree Concept [6]

The attribute selection [16] can be done by the Gini Index process. Gini index is a metric that measure how often the misclassification happened [15]. In this process, the smallest Gini Index value will be selected and be the root node or internal node. As it said in [15], (1), (2), and (3) can be used to determine Gini Index.

$$Gini\ (D) = 1 - \sum_{i=1}^{m} P_i^2 \quad (1)$$

$$Gini_A(D) = \sum_{i=1}^{v} \frac{|D_j|}{D} Gini(D_i) \quad (2)$$

$$Gini\ Gain\ (A) = Gini(D) - Gain_A(D) \quad (3)$$

According to [16], there are five steps to make a decision tree using Gini Index. First is determine the class or label which will be the root in the tree using (1). All the lable's probability with the constraint that has determined will be calculated for the probability and squared. The second is calculate the Gini Index on every attributes or features in the dataset using (2). All the label's probability, in general, will be multiplied by each column of the Gini Index and calculated.

The third is to choose the lowest Gini Index. The feature with the lowest Gini Index will be the root of an internal node in the decision tree. Then, in each branch, do the recursive way from the first step until the leaf or the Gini Index on the branch is zero. Then the last thing is to calculate the Gini Gain with (3) to determine the difference from (1) using Gini Index from each first branch.

F. Random Forest Classifier

Random Forest is an algorithm of machine learning that has ability to do the regression or classification task [17]. This algorithm consists of many decision trees which randomly selected from the subset of the training set. The classification of Random Forest is the accumulation of the votes which decision trees did. The process can be seen in Fig.5.

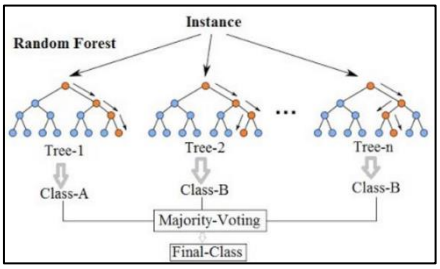

Fig. 5. Random Forect Classifier Structure [9]

According to [18], bagging or bootstrap aggregation is the common technique used for Random Forest Algorithm. The bagging technique is used to reduce the variance of the prediction function which has been estimated. This method is considered effective for the data which has big variance and procedure which has a low bias like trees. In the case of classification, each tree will produce the prediction result which will calculate the majority result or in other words majority vote [18].

The tree-making process [19] can be done by Gini or Entropy. Gini is preferred for the attribute which has continuity, meanwhile, entropy is commonly used for a discrete attribute that exists on each label. The tree-making process can be seen in (4), (5), and (6) which is equal to the decision tree formula [20]. A decision tree can be made recursively based on the tree amount which has been determined. [19] also said that commonly the optimal tree amount made for the classification process is $\sqrt{p}$ and for regression is $\frac{p}{3}$ which p is the predictor amount that will be used. The output of each tree for the classification process will be submitted and there will be a majority vote process like (4) [18].

$$\hat{C}_{rf}^{B}(x) = majority\ vote\{\hat{C}_b(x)\}_1^B \qquad (4)$$

IV. Result and Discussion

A. Dataset

In work conducted in this paper, we use user feedbacks from another research [9]. There is a total of 553 feedbacks, which is contains of 259 positive, 241 negative, and 53 neutral feedbacks.

B. Performance Evaluation

There are several scenarios are conducted to determine the best configuration for classifying the feedbacks' sentiment. An experiment with two training-testing ratios, 70:30 and 80:20, is conducted. First we try to use the feedback as it is (Scenario 1). Furthermore, we try to upsample the neutral feedbacks (Scenario 2) and use the GridSearchCV library to find the best hyperparameter (Scenario 3).

In each scenario, we used different embedding dataset to make the FastText pre-trained model, such as user feedbacks of BYU and Tokopedia app from Playstore (200 & 300 dimension), and SentimentAppReview dataset itself. We also used the original FastText pre-trained model from its official website as the feature extraction.

In the experiment following Scenario 1, the best performance is reached when the embedding dataset is BYU (200 and 300 dimension) and the training-testing ratio is 80:20. The accuracy is 70,27%, the precision is 80%, the recall reaches 54%, and the F1 score is same as the recall. This scenario can result the precision, recall, and F1 score equally. In all scenarios, the performance to detect neutral feedbacks is lower than the others because it is unbalanced. The neutral feedbacks from the dataset is just 53 of 553 row data.

In general, if there is some addition on the test set and subtraction on a train set, the accuracy possibly increases. But, in this research, the 80:20 ratio is better than 70:30. Not only on accuracy, but also the precision, recall, and F1. Apart from the difference of train and test set ratio, the FastText pre-trained model which has chosen just give a small impact on the evaluation result.

Both 80:20 and 70:30, in general, seem hard to detect the neutral label. This is caused by the lack of neutral labels in the dataset. It just 53 if 553 rows. That is why the classifier model is hard to detect the neutral label due to the underfitting issue.

From the requirement engineering side, user feedback can be used to learn the needs, complaints, and user reviews. The negative feedbacks generally contain information that can be useful for future releases, like the bug report. The positive feedbacks usually contain the expression of thanks and user's satisfaction when they use the feature. So, it is important to detect the feedback which has negative or positive sentiment.

The precision calculates the prediction for the true positive to all positive label, that is why the precision score is the ideal measurement when the false positive has the high cost. In this case, false positive happens when feedback whose label is negative but predicted as positive. Meanwhile, the false-negative happens when feedback whose label is positive but it predicted as negative.

In Scenario 2, we try to up-sample the neutral feedbacks. The result is no one of the accuracy reached 70%. The best performance is reached when we used Tokopedia (200 dimension) as embedding dataset and the training-testing ratio is 70:30. The accuracy is 69,63%, the precision is 76%, the recall is 63%, and the F1 score is 64%. According to the result

of this scenario, up-sample can increase the performance from the precision, recall, and F1 score side, but it decreases the accuracy.

Scenario 2 can produce the best result when using Sentiment App Review (200 dimensions) as the FastText pre-trained model and with the ratio of 80:20. In Table 1, we can see that the result of the training set is nearly perfect. There is still misclassification on the neutral label due to the lack of the neutral label data on the dataset and the up-sampling process which is done is not too big. Meanwhile, the data which has positive and negative label relatively give the precise prediction result because the amount of both is quite bigger and balanced.

TABLE II. THE BEST MODEL PERFORMANCE OF SCENARIO 2

Real Label \ Predicted	Negative	Neutral	Positive
Negative	192	0	1
Neutral	0	48	0
Positive	0	0	207

Based on the result and analysis of Scenario 2, upsampling is proven to increase the performance of the precision, recall, and F1 score, but it decreases the accuracy score even it is not too significant.

In Scenario 3, we try to use GridSearchCV to find the best hyperparameter for the Random Forest Classifier. GridSearchCV is usually used to tune the hyperparameter. The method that used in this case is Cross-Validation. It is the statistical method to evaluate the model or algorithm where the data is separated into two subsets, which are training and test. Like the train_test_split, but in Cross-Validation, the data is split into the fold which has been determined. In each Fold, there are train and test sets. Its process is iteratively done until all data from the dataset is included in the fold.

With the help of the GridSearchCV library, the Random Forest Classifier can tune its hyperparameter automatically based on the initial parameter that has been determined to find the best hyperparameter that can produce the best result. The hyperparameter tuned by the GridSearchCV in this case is the amount of leaf, tree, and max features. The tuning technique is done by initializing the input parameters in an array which will be tried on one by one by the library.

In our case, the best parameter for the Random Forest Classifier is 81 for the n_trees, 15 for the n_leafs, and log 2 for the max_features. The best result is reached when BYU (200 and 300 dimension) user feedbacks are used as the embedding dataset and the training-test ratio is 70:30. The accuracy is 71.68, the precision is 48%, the precision is 53%, and the F1 is 50%. This result if compared with Scenario 1, it is lower than Scenario 1. This scenario cannot detect the neutral feedbacks better than Scenario 1.

V. CONCLUSION AND FUTURE WORKS

In this research, we used FastText pre-trained model as the feature extraction and Random Forest Classifier to classify the sentiment of the user feedbacks. With this approach, we used the SentimentAppReview dataset from [12] which then is classified into positive, neutral, and negative label.

In conclusion, the implementation of word embedding to classify user feedback using Random Forest Classifier is successfully done. FastText pre-trained model which is used in this research is made of the user feedback dataset from the Tokopedia and ByU App at the Google Playstore.

Based on the results before, the best result is reached when we used BYU user feedbacks dataset as embedding dataset to make FastText pre-trained model as the feature extraction and the training-testing ratio is 80:20. The accuracy is 70,27%, the precision is 80%, the recall is 54%, and F1 score is 54%.

In the future work, we hope that there is a larger scope of the dataset so it maybe increases the performance with the same method. The more data we can get, hopefully, the better the result that we can get.

Furthermore, to make the more various results we can try different up-sampling or down-sampling methods. Hopefully, with this variance of methods, the result of the experiments can be more variance too.

Trying another word embedding library or feature extraction methods such as Word2Vec and GloVe is not a bad idea. With the help of the Gensim library, that thing can be made easier and hopefully give better performance.

Trying another cross-validation method or library also can be a good thing. For example, using the RandomizedSearchCV library to do the cross-validation method. With this, maybe the hyperparameter tuning can be more variative and give a better result.

If this classification method is combined with synonym extraction, we think it will increase the performance significantly. It will classify the word which has the same meaning with other word into one class more accurately. It will decrease the misclassification also.

REFERENCES

[1] A Van Lamsweerde (2009). Requirements engineering : from system goals to UML models to software

specifications. Chichester, England ; Hoboken, Nj: John Wiley, Cop.

[2] Morales-Ramirez, I. (2013). On Exploiting End-User Feedback in Requirements Engineering. [online] Available at: https://www.researchgate.net/publication/237064827_On_Exploiting_End-User_Feedback_in_Requirements_Engineering

[3] Gao, C., Zeng, J., Lyu, M.R. and King, I. (2018). Online app review analysis for identifying emerging issues. Proceedings of the 40th International Conference on Software Engineering - ICSE '18.

[4] Palomba, F., Linares-Vasquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D. and De Lucia, A. (2015). User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME).

[5] Zhang, L., Wang, S. and Liu, B. (2018). Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4).

[6] Serrano-Guerrero, J., Olivas, J.A., Romero, F.P. and Herrera-Viedma, E. (2015). Sentiment analysis: A review and comparative analysis of web services.Information Sciences, 311, page.18–38.

[7] Fiarni, C., Maharani, H. dan Pratama, R., 2016, May. Sentiment analysis system for Indonesia online retail shop review using hierarchy Naive Bayes technique. In Information and Communication Technology (ICoICT), 2016 4th International Conference on (page. 1-6). IEEE.

[8] Gupte, A., Joshi, S., Gadgul, P. and Kadam, A. (2014). Comparative Study of Classification Algorithms used in Sentiment Analysis. International Journal of Computer Science and Information Technologies, 5(0975–9646), page.6261, 6264.

[9] Vora, P., Khara, M. and Kelkar, K. (2017). Classification of Tweets based on Emotions using Word Embedding and Random Forest Classifiers. International Journal of Computer Applications, 178(0975 – 8887).

[10] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. [online] Tersedia di:https://arxiv.org/pdf/1301.3781.pdf [Accessed 6 Feb. 2020].

[11] Sommerville, I. (2016). Software engineering. Harlow: Pearson Education.

[12] Pamungkas, E.W., Putri and Prasetyo, D.G. (2017). Word Sense Disambiguation for Lexicon-Based Sentiment Analysis. In: Proceedings of the 9th International Conference on Machine Learning and Computing. ACM, pp.442–446.

[13] Abdullah, A. (2018). Word Embedding. [online] Available at: https://rpubs.com/ahmadhusain/wordembedding [Accessed 8 Jan. 2020].

[14] Rehurek, R. (2019). Gensim: Topic Modelling For Humans. [online] Radimrehurek.com. Available at: https://radimrehurek.com/gensim/auto_examples/tutorials/run_fasttext.html#sphx-glr-auto-examples-tutorials-run-fasttext-py [Accessed 6 Feb. 2020].

[15] Mitchell, T.M. (2017). Machine learning. New York: Mcgraw Hill.

[16] Han, J., Kamber, M. and Pei, J. (2012). Data Mining Concepts and Techniques. 225 Wyman Street, Waltham, MA, 02451, USA: Morgan Kaufmann Publishers.

[17] Eletter, S., Yasmin, T., Elrefae, G., Aliter, H., & Elrefae, A. (2020). Building an Intelligent Telemonitoring System for Heart Failure: The Use of the Internet of Things, Big Data, and Machine Learning. 2020 21st International Arab Conference on Information Technology (ACIT). https://doi.org/10.1109/acit50332.2020.9300113

[18] Hastie, T., Tibshirani, R. and Friedman, J. (2009). The elements of statistical learning, second edition: data mining, inference, and prediction. New York: Springer

[19] Sieling, G (2014). Decision Trees: "Gini" vs. "Entropy" criteria. Minnesota.

[20] Breiman, L. (2001). Random Forests. [online] 45. Tersedia di: stat.berkeley.edu/forests_V3.1.pdf [Accessed 8 Mar. 2020