Proceeding of International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2015), Palembang, Indonesia, 19 -20 August 2015

The Influence of Stemming on Indonesian Tweet Sentiment Analysis

Ahmad Fathan Hidayatullah
Department of Informatics, Universitas Islam Indonesia, UII, Yogyakarta, Indonesia
fathan@uii.ac.id

***Abstract*—Stemming has commonly used in some research about text mining, information retrieval, and natural language processing. However, there is an indication that stemming does not deliver significant influence toward accuracy in text classification. Hence, this research attempts to investigate the influence of the stemming process on Indonesian tweet sentiment analysis. Furthermore, this work examines about the difference effect between two conditions by involving stemming and without involving stemming on pre-preprocessing task. The experiments show that the accuracy difference for SVM using stemming in pre-processing acquired 0.67% and 1.34% higher than pre-processing without stemming, whereas, Naive Bayes obtained 0.23% and 1.12%. Finally, this research proves that stemming does not raise the accuracy either using SVM or Naive Bayes algorithm.**

Keywords—pre-processing; stemming; sentiment analysis; tweet classification

I. Introduction

Pre-processing is very important and critical in text mining and information retrieval because of the unstructured data from natural language (Torunoğlu, Çakırman, Ganiz, Akyokuş, & Gürbüz, 2011). Pre-processing has a purpose to set up and clean the text data before further processing in classification [2]. In pre-preprocessing, there is a step called stemming, which is a basic step in handling textual data [3]. However, there is an indication that stemming does not hand over significant influence toward accuracy in text classification [4] [5]. Hence, this research attempts to investigate the influence of stemming on Indonesian tweet sentiment analysis.

Stemming has commonly used in some research about text mining, information retrieval, and natural language processing. The purpose of stemming is to alleviate various gramatical form and find the root of words by removing affixes and suffixes [6] [7] [8].

The words in Bahasa Indonesia have the special and complex morphological structure compared with the words in other languages. Generally, they consist of both inflectional and derivational structure [9]. Inflectional is a collection of suffixes which does not alter the form and does not affect the meaning of the root word. Derivational structure in Bahasa Indonesia can be composed of prefixes, suffixes, and also a couple of combination of the two.

This research uses Nazief and Adriani algorithm in stemming process to obtain the root of the words. This algorithm uses morphological rules to remove affixes, including prefixes, suffixes, infixes (insertions) and confixes (composite between prefixes and suffixes) [10].

The data in this research are gained from tweet which contain the sentiment about politician in Indonesia. This work examines those data and investigate what will be happened when stemming does not carried out in pre-processing task, whether stemming will increase the accuracy and give some benefits on classifying Indonesian tweet or not. If stemming does not give more effect or advantages in Indonesian sentiment analysis, then stemming task can be removed from pre-processing task. In the other hand, suppose that stemming is still giving advantages such as increasing the accuracy, then stemming must be carried out and continue to develop better stemming process in pre-processing task for better attainment.

The rest of this paper is organised as follows. Section 2 describes about several works correlated with stemming. In section 3, there is a proposed method for this work. Section 4 explains the result and discussion of this research. Finally, the conclusion of this work is described in Section 5.

II. Related Works

A lot of research about stemming has been carried out before. Jivani [7] tried to compare the English stemming algorithms and the difference between stemming and lemmatizing. This research discussed about the different methods of stemming and their comparisons in terms of usage, the benefits and also the restrictiveness. Based on the experiment, there are a lot of similarity between the stemming algorithms and they are good enough to be used on text mining, NLP, or information retrieval.

Asian, et al. [10] compared the performance of stemming algorithm as well, but this research focused on Indonesian stemmings algorithm. Based on the result, good stemmer is complex. It also need thorough combination of several features, such as support for complex, morphological rules, progressive stemming of words, dictionary checks after each step, trial and error combinations of affixes, and recording support after prefix removal. This research concluded that Nazief and Adriani approach should be performed for Indonesian stemming.

Different to [7] and [10], Toman, et. al [4] focused on the comparison of various lemmatization and stemming algorithms which are regularly utilized in Natural Language Processing (NLP). Moreover, this research also worked through the influence of the word normalization on general classification task using two datasets, English and Czech dataset. However, this research summarized that the word normalization does not affect significantly in text classification.

Another research that also examine the influence of stemming done by Wahbeh, et. al [5]. This research, investigated the effect of stemming as part of the pre-processing tasks on Arabic text classification. Based on the experiment using Support Vector Machine (SVM), applying text classification without stemming has reached 87.79% and 88.54%. On the contrary, stemming has decreased the accuracy, where the two test modes using SVM dropped down to 84.49% and 86.35%. However, Basnur and Sensuse [11] have difference result concerned with the effect of stemming toward accuracy. The research classified news article in Indonesian Language using ontology by observing two aspects, stopwords and stemming. The research concluded that using stopwords and stemming in classifying document could enhance the accuracy.

III. Proposed Method

The proposed method in this paper consists of four tasks such as text pre-processing, term weighting, classification method, and performance evaluation. Fig. 1 shows the detail experimental design in this work.

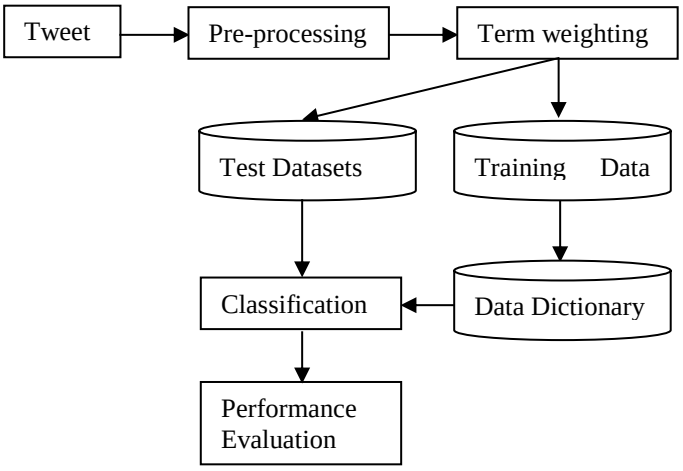

Fig. 1. Experimental Design

A. Dataset

The tweet dataset used in this research is almost the same with the dataset that used by Hidayatullah [12]. The data consist of tweets about Indonesian public figure sentiment, especially a politician. This work examines 1345 tweets which labelled manually into positive and negative. Table I shows that the number of positive tweet is 743 and negative is 602.

TABLE I. DETAIL NUMBER OF TWEET POLARITY

Tweet Polarity	Quantity
Positive	743
Negative	602

B. Text Pre-processing

There are some tasks in pre-processing proposed in this research such as removing URLs, changing emoticon, removing special characters of Twitter, removing symbols, tokenization, removing public figure name, case folding, changing slangword, stemming, removing stopword, and concatenating negation.

1. Removing URLs task handles the URLs that sometimes found in tweet, for example http://www.website.com, name@address.com, etc.
2. Changing emoticon step replaces the emoticons that present in tweet by transforming the emoticon into representative words. The conversion could be seen on Table II.

TABLE II. EMOTICON CONVERSION

Emoticon	Conversion
:) :-) :)) :-)) =) =))	Senyum (smile)
:D :-D =D	Tawa (laugh)
:-(:(	Sedih (sad)
;-) ;)	Berkedip (wink)
:-P :P	Mengejek (stick out tongue)
:-/ :/	Ragu (hesitate)

-
3. Twitter has some special characters such as hashtag (#hashtag), username or target (@username), and RT that refers to retweet from users. Removing special characters of Twitter is done by removing those characters that mentioned before.
4. Tweet usually also contains symbols or numbers (e.g. !, #, $, *, 1234, etc.), therefore those symbols and numbers will be removed.
5. Tokenization is the methodology of separating a stream of text into phrases, words, symbols, or other significant components called tokens [13].
6. The public figure name which appears in the tweet will be omitted in this step.
7. Case folding is the process to transform words into similar form, uppercase or lowercase. In this research, all letters are transformed into lowercase.
8. There are many slangwords used when people post their tweet. So, slangword will be changed into standard word based on dictionary.
9. Stemming is used to reduce the affixes and suffixes in the word.

10. Removing stopword task removes the stopword (i.e. 'dan' (and), 'or' (atau), etc.) that occur in the tweet.
11. This work also proposes the task to concatenate negation. This task recognises negation in tweet for example when there is a term *'tidak'* (not) then the word *'tidak'* will be concatenated with the next word.

C. *Nazief and Adriani's Algorithm*

The steps of Nazief and Adriani's Algorithm are described as follows [10] :

1. Firstly, find the current word in the dictionary. The word will be considered to be a root word if it found and the process stops.
2. The inflection suffix ("-lah", "-kah", "-ku","-mu", or "-nya") will be removed. If it succeeds and the suffix is a particle ("-lah" or "-kah"), then remove the inflectional possessive pronoun suffix ("-ku", "-mu", or "-nya").
3. Remove the derivation suffix ("-i" or "-an"). If this succeeds, then go to the step 4. Otherwise, if step 4 does not succeed, do these steps :
   a. If "-an" was removed, and the last letter of the word is "-k", then the "-k" is also removed and try again Step 4. If that fails, Step 3b will be carried out.
   b. The removed suffix ("-i", "-an", or "-kan") is brought back.
4. Remove the derivation prefix, as "di-","ke-","se-","me-","be- ","pe", "te-" by attempting these steps :
   a. If a suffix was eliminated in Step 3, then check the dissalowed prefix-suffix combinations which is listed in Table III.

TABLE III. DISSALLOWED PREFIX-SUFFIX COMBINATIONS

Prefix	Dissallowed Suffixes
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
te-	-i, -kan
se-	-an

   b. The algorithms returns if the latest word appropriates with any previous prefix.
   c. The algorithms returns if three prefixes have previously been deleted.
   d. The prefix type is recognised by one of these following actions :
      i. If the prefix of the word is "di-", "ke-", or "se-", then the prefix type is "di", "ke", or "se" successively.
      ii. When the prefix is "te-", "be-", "me-", or "pe-", an extra process of extracting character sets to determine the prefix type is needed. For example, the word "terlambat" (late) will be stemmed. After the prefix "te-" removed to obtain "-rlambat", the first collection of characters are extracted from the prefix as initiated by the "Set 1" rules in Table IV. This example the letter after the prefix "te-" is "r" and this is appropriate with the first five rows of the table. The letter "r" is followed by "l" and match with third to fifth rows (Set 2). The next after the letter "l" is "ambat" and this is exactly with the fifth row (Set 3), so the prefix type in the word "terlambat" is "ter-".

TABLE IV. DETERMINING THE PREFIX TYPE FOR WORDS PREFIXED WITH "TE–"

Following Characters				Prefix Type
Set 1	**Set 2**	**Set 3**	**Set 4**	
"-r-"	"-r-"	-	-	None
"-r-"	Vowel	-	-	Ter-luluh
"-r-"	not ("-r-" or vowel)	"-er-"	vowel	ter
"-r-"	not ("-r-" or vowel)	"-er-"	not vowel	none
"-r-"	not ("-r-" or vowel)	not "-er-"	-	ter
not (vowel or "-r-")	"-er-"	vowel	-	none
not (vowel or "-r-")	"-er-"	not vowel	-	te

      iii. The algorithm returns when the first two characters do not match with "di-", "ke-", "se-", "te-", "be-", "me-", or "pe-".
   e. If the prefix type is "none", then the algorithm returns. For not "none" prefix type, it is found in Table V and the prefix will be removed from the word.

TABLE V. DETERMINING THE PREFIX FROM THE PREFIX TYPE

Prefix type	Prefix to be removed
di	di-
ke	ke-
se	se-
te	te-
ter	ter-
ter-luluh	ter-

   f. Step 4 will recursively endeavoured when the root word has not been found until the root word found.
   g. Perform the recoding step which is depending on the prefix type. It is only shown the prefix type "ter-luluh" in Table IV and V. In this case, the letter "r-" is added to the word after removing the prefix "ter-". If the new

word is not found in the dictionary, then Step 4 is carried out again. The "r-" is removed and "ter-" restored when the root word is still not found. The prefix is is set to "none" and the algorithm returns.

h. The algorithm will return the authentic word after failed in all steps.

D. Term Weighting

Term weighting represents documents as vectors in the vector space model [14]. In this paper, we conduct two methods to represent documents, there are TF (Term Fequency) and TF-IDF (Term Frequency-Inverse Document Frequency).Term frequency is the standard idea of frequency in corpus-based natural language processing (NLP). It calculates the quantity of times that a type (term/word/n-gram) shows up in a corpus [15]. The term frequency of a term t in a document d can be utilized for record particular weighting and denoted as $tf_{t,d}$. It is just a measure of a term hugeness inside a document [16].

TF-IDF is a method which combines both TF and IDF to determine the weight of a term [14]. The TF-IDF weight scheme of a term t in a document d given by [17],

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \quad (1)$$

The weight of term t in document d is denoted as $tf_{t,d}$, whereas idf_t is inverse document frequency of term t which derived from

$$\text{idf}_t = \log \frac{N}{\text{df}_t} \quad (2)$$

In the equation 2, the number of all document represented as N and df_t is the quantity of document d which contains term t.

E. Classification Method

This research uses Naive Bayes and Support Vector Machine which are the most famous text classification method [18] [19]. Furthermore, both of those methods also have good performance in classification.

1) Multinomial Naive Bayes

This research uses Multinomial Naive Bayes to classify the tweet data. Multinomial Naive Bayes is a probabilistic learning approach [17]. The likelihood of a document *d* being in class *c* is processed as,

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c) \quad (3)$$

$P(t_k|c)$ is the probability of term t_k occuring in a document of class *c* whereas P(c) is the prior probability of a document occurring in class *c*. Variable n_d is the number of token in document *d*.

The best class in Naive Bayes Classification is the most probability that procured from *Maximum a posteriori (MAP)* class c_{map}:

$$c_{map} = \underset{c \in C}{\text{argmax}} \hat{P}(c|d) = \underset{c \in C}{\text{argmax}} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c) \quad (4)$$

The value of $\hat{P}(c)$ is calculated using equation (5) by dividing the number of document in class *c* (N_c) with all number of document in training data (*N'*).

$$\hat{P}(c) = \frac{N_c}{N'} \quad (5)$$

2) Support Vector Machine (SVM)

SVM finds a hyperplane with the highest possible margin to separate between two categories [20]. The hyperplane is represented by vector w. The Support Vectors are the items which determine the margin (H1 and H2), shown by Fig. 2.

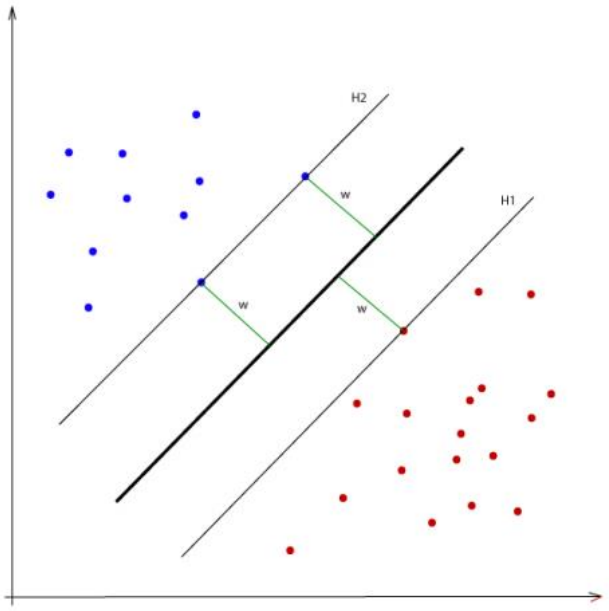

Fig. 2. Separating Hyperplane in SVM

F. Evaluation

The purpose of evaluation is to measure how good the system works [11]. In this research we choose confusion matrix model for two classes prediction shown by Table III.

TABLE VI. CONFUSION MATRIX FOR TWO CLASSES PREDICTION

		Actual Class	
		Class-1	**Class-2**
Predicted Class	Class-1	True positive (TP)	False negative (FN)
	Class-2	False positive (FP)	True negative (TN)

The confusion matrix above is used to calculate the accuracy of the proposed methods. Accuracy is the total validity of the model that calculated as the sum of true classifications divided by

the total number of classifications. The formula to obtain accuracy is described below,

$$\text{Accuracy} = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (6)$$

-

G. Experiment

This research proposes two different approaches in pre-processing task. The first approach uses stemming in pre-processing steps and the second does not use stemming. The stemming rules in this research is using Nazief and Adriani approach based on dictionary. Fig.3 shows the two different approaches in pre-processing task.

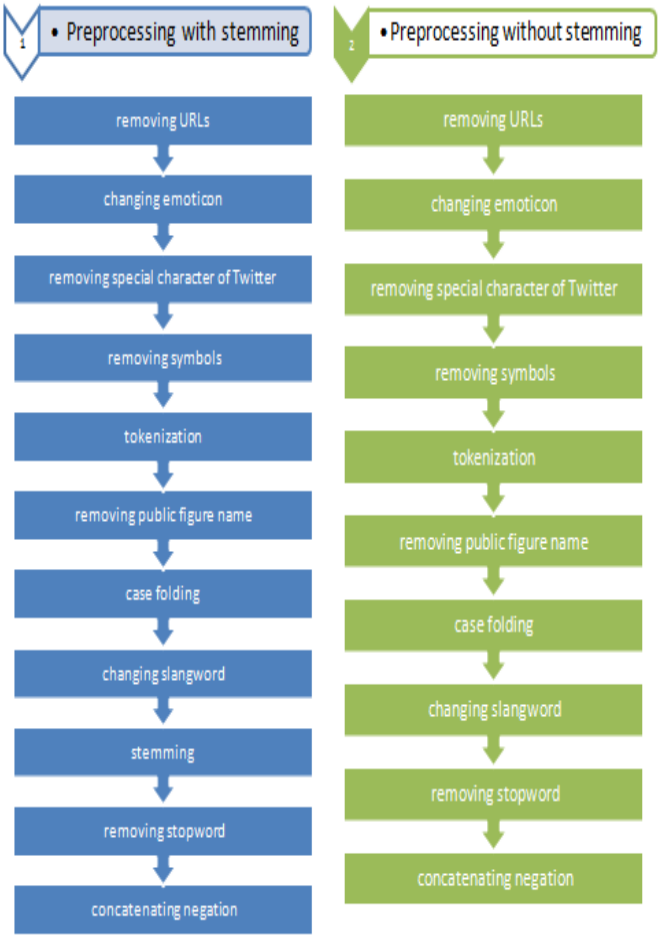

Fig. 3. Two Different Approaches in Pre-processing

To examine the influence of stemming, there are four combinations such as TF-IDF with stemming, TF-IDF without stemming, term frequency (TF) with stemming, and term frequency (TF) without stemming. The classification methods used in this research are SVM and Naive Bayes. In addition, rapidminer is used in this work to analyse the data. The main process in rapidminer shown by Fig. 4 below.

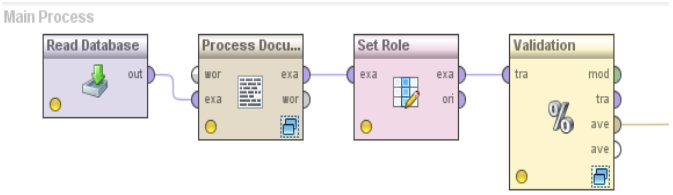

Fig. 4. Main Process in Rapidminer

There are four steps carried out using repidminer such as read database, process document, set role, and validation. Read database retrieves tweet data from database. Then, process document generates word vectors from string attributes. Set role is used to change the attribute role (e.g. regular, special, label, id, etc). Validation randomly splits up the data into training and test set, then evaluates the model. The Validation process using SVM shown by Fig. 5.

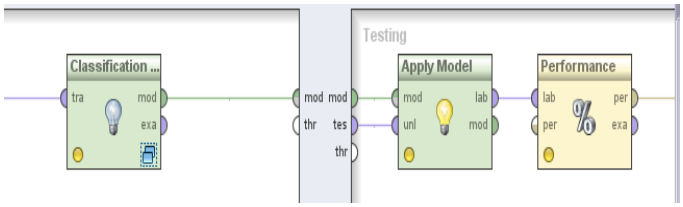

Fig. 5. Validation Process Using SVM

The validation process using Naive Bayes shown by Fig. 6.

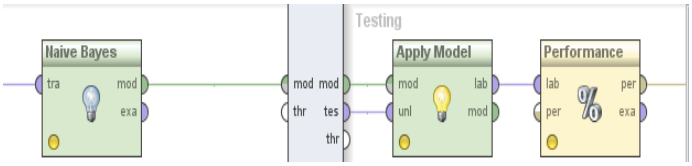

Fig. 6. Validation Process Using Naive Bayes

Furthermore, the model will be applied using apply model then measure the performance.

IV. Result and Discussion

The experiments using rapidminer shows that stemming in pre-processing task does not enhance the accuracy. The accuracy achieved 87.72% when stemming performed for Naive Bayes combined with TF-IDF. In the other hand, the accuracy increased up to 88.84% when stemming does not included in pre-processing. In addition, the accuracy using Naive Bayes and term frequency is 91.96% for pre-processing with stemming, and 92.19% for pre-processing without stemming.

The accuracy gained from SVM and TF-IDF when stemming does not performed is 93.75%. On the contrary, the accuracy gained 92.41% when stemming carried out in pre-processing. Furthermore, the accuracy using SVM and term frequency attained 93.53% for pre-processing without stemming, and 92.86% for pre-processing which involved stemming. The clear result of this research shown by Fig.7.

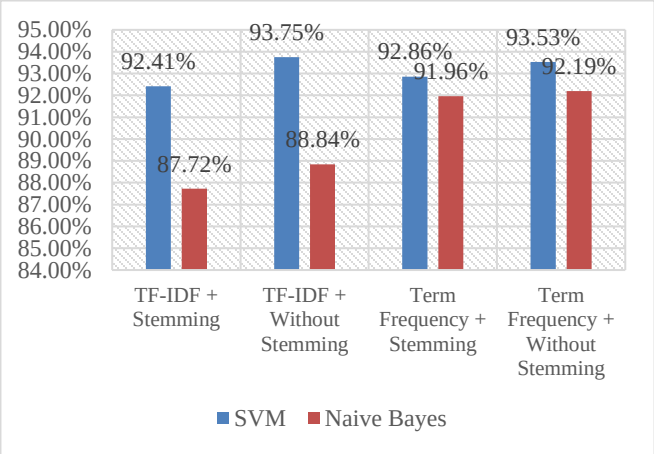

Fig. 7. Accuracy Results

V. Conclusion and Future Works

This research discussed about the influence of stemming on Indonesian tweet sentiment analysis. The investigation of this work has compared between two pre-processing steps, the first pre-processing steps involved stemming and the other one does not involve stemming. Based on the experiments, this work concluded that stemming does not give significant influence on Indonesian tweet sentiment analysis. Stemming task does not raise the result accuracy either using SVM or Naive Bayes algorithm. However, the accuracy difference between pre-processing which involve stemming and does not involve stemming is not too high. The accuracy difference for SVM using stemming in pre-processing acquired 0.67% and 1.34% higher than pre-processing without stemming, whereas, Naive Bayes obtained 0.23% and 1.12%.

For the future works, it is better to use data which have more variation, not only tweet data which talk about politician, but also about sport, commodity review, consumer satisfaction, etc. Moreover, it is also important to combine with another new pre-processing technique in order to enhance the result accuracy.

References

[1] D. Torunoğlu, E. Çakırman, M. C. Ganiz, S. Akyokuş and M. Z. Gürbüz, "Analysis of Preprocessing Methods on Classification of Turkish Texts," in *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on. IEEE*, 2011.

[2] E. Haddi, X. Liu and Y. Shi, "The Role of Text Pre-processing in Sentiment Analysis," *Procedia Computer Science*, vol. 17, pp. 26-32, 2012.

[3] B. Sarumathi and R. Jayanthi, "Identification of Legitimate Words for Text pre-processing Using K-Means Clustering Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 1, pp. 2903-2907, 2014.

[4] M. Tomana, R. Tesara and K. Jezeka, "Influence of Word Normalization on Text Classification," in *Proceedings of InSciT (2006)*, 2006.

[5] A. Wahbeh, M. Al-Kabi, Q. Al-Radaideh, E. Al-Shawakfa and I. Alsmadi, "The Effect of Stemming on Arabic Text Classifiation:An Empirical Study," *International Journal of Information Retrieval Research*, vol. 1, no. 3, pp. 54-70, 2011.

[6] B. Liu, Web Data Mining Exploring Hyperlinks, Contents , and Usage Data, Springer, 2007.

[7] M. A. G. Jivani, "A Comparative Study of Stemming Algorithms," *Int. J. Comp. Tech. Appl*, vol. 2, no. 6, pp. 1930-1938, 2011.

[8] C. Ramasubramanian and R. Ramya, "Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 12, pp. 4536-4538, December 2013.

[9] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia," 2003.

[10] J. Asian, H. E. Williams and S. Tahaghoghi, "Stemming Indonesian," in *Proceedings of the 28th Australasian conference on Computer Science*, 2005.

[11] P. W. Basnur and D. I. Sensuse, "Pengklasifikasian Otomatis Berbasis Ontologi untuk Artikel Berbahasa Indonesia," *MAKARA Journal of Technology*, vol. 14, no. 2, 2010.

[12] A. F. Hidayatullah, "Analisis Sentimen dan Klasifikasi Kategori Terhadap Tokoh Publik Pada Data Twitter Menggunakan Naive Bayes Classifier," 2014.

[13] T. Verma, Renu and D. Gaur, "Tokenization and Filtering Process in RapidMiner," *International Journal of Applied Information Systems (IJAIS)*, vol. 7, no. 2, pp. 16-18, 2014.

[14] V. Srividhya and R. Anitha, "Evaluating Preprocessing Techniques in Text Categorization," *International Journal of Computer Science and Application Issue*, pp. 49-51, 2010.

[15] M. Yamamoto and K. W. Church, "Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in A Corpus," *Computational Linguistics*, vol. 27, no. 1, pp. 1-30, 2001.

[16] S. T. Wu, "Knowledge Discovery Using Patern Taxonomy Model in Text Mining," 2007.

[17] C. Manning, P. Raghavan and H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2009.

[18] D. Li Guo, D. Peng dan L. Ai Ping, "A New Naive Bayes Text Classification Algorithm," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 2, pp. 947-952, February 2014.

[19] P. Y. Zhang, "A HowNet-based Semantic Relatedness Kernel for Text Classification," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 4, pp. 1909-1915, 2013.

[20] C. Horn, "Analysis and Classification of Twitter Messages," 2010.

[21] A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, 2010.

[22] X. Hu, J. S. Downie and A. F.Ehmann, "Lyric Text Mining in Music Mood Classification," in *10th International Society for Music Information Retrieval Conference*, 2009.

[23] B. Liu, "Sentiment Analysis: A Multi-faceted Problem," *IEEE Intelligent Systems*, 2010.