TELKOMNIKA, Vol.14, No.2, June 2016, pp. 665~673
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v14i1.3113

Analysis of Stemming Influence on Indonesian Tweet Classification

Ahmad Fathan Hidayatullah*[1], Chanifah Indah Ratnasari[2], Satrio Wisnugroho[3]
Department of Informatics, Universitas Islam Indonesia,
Jl.Kaliurang km 14.5 Sleman Yogyakarta Indonesia, Telp. (0274) 895297/Fax. (0274) 895007
*Corresponding author, e-mail: fathan@uii.ac.id[1], chanifah.indah@uii.ac.id[2],
wisnugrohosatrio@gmail.com[3]

Abstract

Stemming has been commonly used by some researchers in natural language processing area such as text mining, text classification, and information retrieval. In information retrieval, stemming may help to raise retrieval performance. However, there is an indication that stemming does not hand over significant influence toward the accuracy in text classification. Therefore, this paper analyzes further research about the influence of stemming on tweet classification in Bahasa Indonesia. This work examines about the accuracy result between two conditions by involving stemming and without involving stemming in pre-processing task for tweet classification. The contribution of this research is to find out a better pre-processing task in order to obtain good accuracy in text classification. According to the experiments, it is observed that all accuracy results in tweet classification tend to decrease. Stemming task does not raise the accuracy either using SVM or Naive Bayes algorithm. Therefore, this work summarized that stemming process does not affect significantly towards the accuracy performance.

Keywords*: stemming, pre-processing, tweet classification, text classification*

Copyright © 2016 Universitas Ahmad Dahlan. All rights reserved.

1. Introduction

Stemming has been commonly used by researchers in natural language processing area such as text mining, text classification, and information retrieval. The purpose of stemming is to obtain the root of words by removing affixes and suffixes. Nevertheless, stemming process in every language is different depends on the formation of words in it. For example, English stemming reduces the words “plays”, “player”, “players”, and “played” to the word “play”. To obtain the root of words in English, there are some stemming algorithm such as Lovins, Dawson, and, Porter. On the other hand, Indonesian language has different characteristics from English. The words in Bahasa Indonesia have the special and complex morphological structure compared with the words in other languages. In Bahasa Indonesia, the words composed of inflectional and derivational structure. Inflectional is a collection of suffixes which does not alter the form and does not affect the meaning of the root word. Derivational structure consists of prefixes, suffixes, and also a couple of combination of the two. In Indonesian language, there are various algorithms for stemming like Vega, Nazief-Adriani, Arifin-Setiono, and Enhanced Confix Stripping Stemmer.

However, some previous research has shown different result regarding to the influence of stemming in text mining. Several research clarified that stemming could enhance the accuracy and the others claimed that stemming did not hand over significant influence toward the accuracy.

Basnur and Sensuse [1] have classified news article in Indonesian Language using ontology by observing two aspects, stop words and stemming. They declared that stemming in classifying text documents could enhance the accuracy. Ramasubramanian and Ramya [2] made an effective pre-processing step using improved stemming algorithm. The research concluded that improved Porter’s stemming with the Spell-check utility has increased the accuracy level of output content. Porter’s stemming algorithm also utilized by [3] to evaluate stemming and stop word techniques in classification problem. This paper revealed that stemming techniques have a significant affect to the size of the feature set with a different sparsity value.

Received January 10, 2016; Revised April 10, 2016; Accepted April 26, 2016

Gaustad and Boume [4] conducted the investigation about the use of stemming for classification of Dutch email texts. This research also used Porter's algorithm for stemming process. It was found that stemming does not consistently improve classification accuracy. Yu [5] evaluated the effect of stemming on classification performance. The research examined whether the overall classification accuracies change significantly after stemming. In addition, it also compared the contribution of individual features before and after stemming toward classification. According to the experiment, the accuracy did not change significantly before and after stemming.

Torunoglu, et al., [6] analyzed the effect of preprocessing methods in text classification on Turkish texts. They concluded that stemming has very little impact on accuracies. Toman, et al., [7] compared between various lemmatization and stemming algorithms using English and Czech datasets to examine the influence of the word normalization on general classification task. This research summarized that lemmatization and stemming in word normalization did not affect significantly in text classification. Wahbeh, et al., [8] examined the effect of stemming as part of the pre-processing tasks on Arabic text classification. Their experiment referred that stemming has decreased the accuracy. Hidayatullah [9] clarified that stemming does not raise the accuracy as well. This research uses Nazief and Adriani algorithm in stemming process to obtain the root of the words. Nazief and Adriani Indonesian stemming algorithm is choosen because it has better accuracy than any other stemming algorithms [10].

This paper addresses the issue of text classification in Bahasa Indonesia. Furthermore, this research conducts further investigation about the influence of stemming on text classification using tweet dataset in Indonesian Language. Moreover, this work examines about the difference effect between two conditions by involving stemming and without involving stemming on pre-preprocessing task. According to previous research by Hidayatullah [9], the number of datasets in this research also increased to obtain more valid result.

The rest of this paper is organized as follows. Section 2 describes the research method in this work. Section 3 explains the result and discussion of this research. Finally, the conclusion of this work is described in Section 4.

2. Research Method

Figure 1 depicts the detail experimental design in this work. In data collection, tweet datasets were gathered using Twitter Search API v1.1. Secondly, pre-processing step is conducted to clean the tweet from noisiness. The third step is feature selection which aims to get the influential feature and remove the uninfluential feature. This research uses two term weighting methods in feature selection task, term frequency and TF-IDF (Term Frequency-Inverse Document Frequency). The tweet datasets will be classified using Naive Bayes Classifier and Support Vector Machine (SVM) method.

Figure 1. Experimental design

Finally, performance evaluation is carried on to evaluate the proposed method. The objective of performance evaluation is to measure how precise the method in classifying text. The confusion matrix model is chosen for two class prediction which can be seen in Table 1.

Table 1. 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:

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \quad (1)$$

2.1. Datasets

Tweet datasets were gained from the previous datasets used by Hidayatullah [9]. Moreover, this research also increase the number of tweet datasets and examined 2000 tweets. Those tweets were labelled manually into two sentiment polarities, positive and negative. The datasets contain 1074 positive tweets and 926 negative tweets as shown in Table 2.

Table 2. Distribution of Tweet Polarity

Tweet Polarity	Quantity
Positive	1074
Negative	926

2.2. Text Pre-processing

This research proposes several steps in text pre-processing, such as:

1. Removing URLs task handles the URLs in tweet, for example http://www.website.com.
2. Changing emoticon step replaces the emoticons that present in tweet by transforming the emoticon into representative words which shown in Table 3.

Table 3. Emoticon Conversion

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

3. Removing special characters of Twitter such as #hastags, @username, and RT (retweet).
4. Removing symbols or numbers (e.g.!, #, $, *, 1234, etc.)
5. Normalize lengthening words, for example the word *'semangaaaaatttt'* will be normailzed into *'semangat'* which means spirit.
6. Tokenization separates a stream of text into parts called tokens.
7. The public figure name which appears in the tweet will be omitted in this step.
8. Case folding transforms words into similar form (lowercase or uppercase).
9. Change slang words into standard word based on dictionary.
10. Stemming is used to reduce the affixes and suffixes in the word.
11. Removing stopword task removes the stopword.
12. Concatenate negation recognizes negation in tweet for example when there is a term *'tidak'* (not) then the word *'tidak'* will be concatenated with the next word.

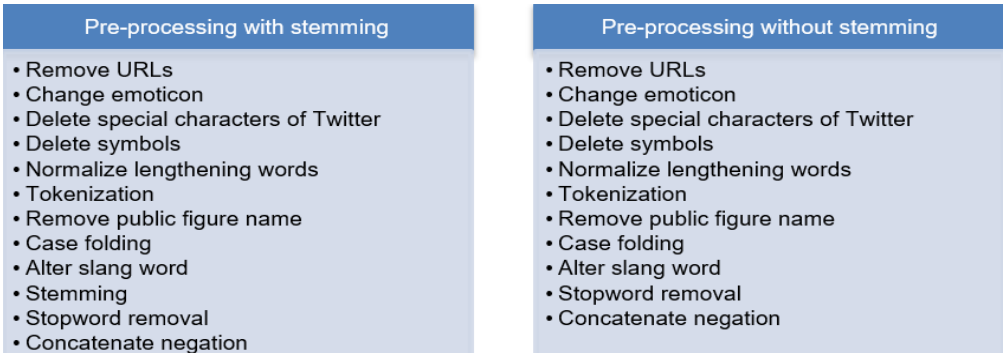

Figure 2. Two different approaches in pre-processing

Two different approaches were conducted to compare the result accuracy between involve stemming and the second do not involve stemming in pre-processing steps. The stemming rules in the first approach utilized Nazief and Adriani algorithm. Figure 2 shows the two different approaches in pre-processing task.

2.3. Nazief and Adriani's Algorithm

The steps of Nazief and Adriani's Algorithm are described as follows [11]:

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 omitted. 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 4.

Table 4. 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 recognized by one of these following actions :
      1) If the prefix of the word is "di-", "ke-", or "se-", then the prefix type is "di", "ke", or "se" successively.
      2) 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 5. 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 5. 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

3) 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 6 and the prefix will be removed from the word.

Table 6. 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 5 and 6. 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 original word after failed in all steps.

2.4. Feature Selection Methods

1. Term 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 [12]. The term frequency of a term *t* in a document *d* can be utilized for record particular weighting and denoted as $tf_{t,d}$.

2. TF-IDF

TF-IDF combines both TF and IDF to determine the weight of a term [13]. The TF-IDF weight scheme of a term *t* in a document *d* given by [14]:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \quad (2)$$

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:

$$idf_t = \log \frac{N}{df_t} \quad (3)$$

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*.

2.5. Classification Method

1. Naive Bayes

Naive Bayes is a probabilistic learning approach [14]. The likelihood of a document *d* being in class *c* is processed as:

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c) \quad (4)$$

$P(t_k|c)$ 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}:

$$c_{map} = \underset{c \in C}{argmax} \hat{P}(c|d) = \underset{c \in C}{argmax} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c) \quad (5)$$

The value of $\hat{P}(c)$ is calculated using Equation (6) by dividing the number of document in class *c* (N_c) with all number of document in training data (*N'*).

$$\hat{P}(c) = \frac{N_c}{N'} \quad (6)$$

2. Support Vector Machine (SVM)

SVM finds a hyperplane with the highest possible margin to separate between two categories. Hyperplanes with larger margin are less likely to overfit the training data. Suppose that, we have datasets $\{x_1, x_2, \ldots, x_n\}$ whereas $y_i \in \{+1, -1\}$ as the class label of x_1. Figure 3 shows the hyperplane with the largest margin, which separate between two classes. The hyperplane can be written as:

$$x_i . w + b = 0 \quad (7)$$

In Equation (7), *w* is a weight vector $\{w_1, w_2, \ldots, w_n\}$ and *b* is an additional weight. According to Figure 3, there are two hyperplanes that define the sides of the margin. Both of hyperplanes can be written as:

$$x_i . w + b \ge +1 \; for \; y_i = +1 \quad (8)$$

$$x_i . w + b \le -1 \; for \; y_i = -1 \quad (9)$$

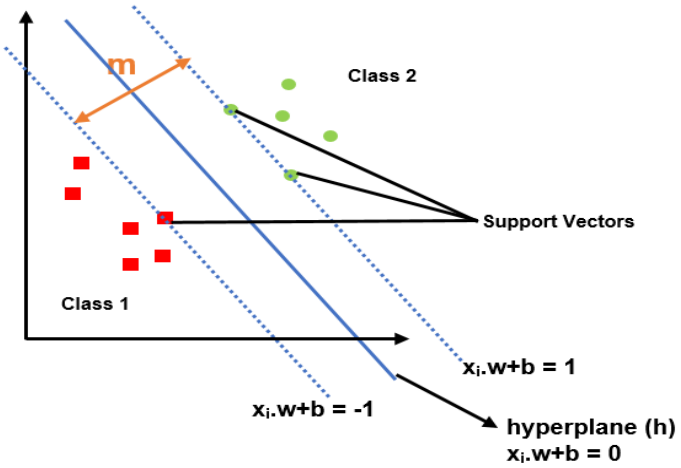

Figure 3. Hyperplane with the largest margin

2.6. Experiment

This work uses Rapidminer 5.3.000 to conduct the experiment. Moreover, this research proposes four combinations in the experiment such as TF-IDF with stemming, TF-IDF without stemming, term frequency (TF) with stemming, and term frequency (TF) without stemming. Each combination will be classified using Naive Bayes Classifier and SVM. The main process using Rapidminer can be seen in Figure 4 below.

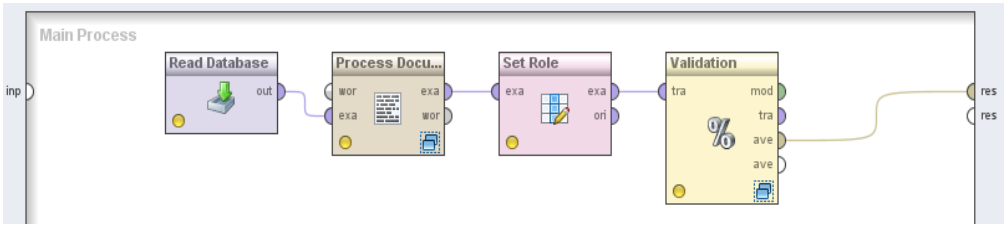

Figure 4. Main Process in Rapidminer

The main process in Rapidminer uses four operators such as read database, process document, set role, and split validation. Read database operator retrieves tweet data from database. Process document operator generates word vectors from string attributes using term frequency or TF-IDF. Set role operator is used to change the attribute role (e.g. regular, special, label, id, etc). Split validation operator randomly splits up the data into training and test set then evaluates the model.

The training and testing process are conducted in the sub process of the split validation operator. Three operators choosen in the sub process, modeling operator, apply model, and performance. In modeling operator, the experiment uses Bayesian Modeling for Naive Bayes method. For SVM method, it uses Support Vector Modeling. Figure 5 depicts the training and testing process using Naive Bayes method.

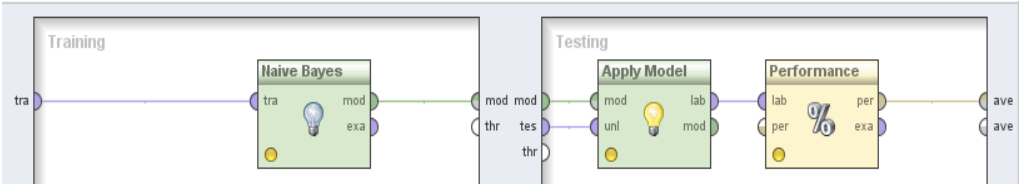

Figure 5. Training and testing process using Naive Bayes

The scenario for training and testing process using SVM can be seen in Figure 6 below.

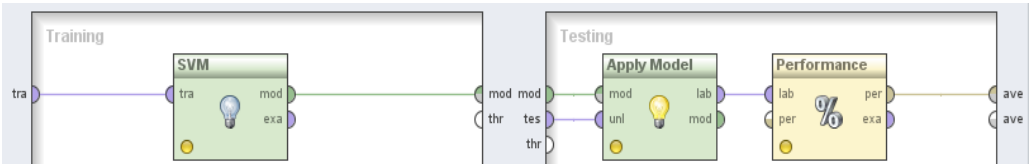

Figure 6. Training and testing process using Naive Bayes

3. Results and Analysis

To examine the influence of stemming in tweet classification, our experiments compare the accuracy based on the number of datasets. The datasets are divided into three different subset size (1500, 1750, and 2000).

The experiments using Naive Bayes and TF-IDF can be seen in Table 7. The accuracy results with stemming are lower than without stemming. However, the accuracy difference between both treatments are not too high with the average 0.86%.

Table 7. Experiments Results Using Naive Bayes and TF-IDF

Number of Datasets	Accuracy		Accuracy Difference
	TF-IDF + Stemming	TF-IDF + Without Stemming	
1500	85.33%	86.00%	0.67%
1750	84.00%	84.57%	0.57%
2000	84.00%	85.33%	1.33%
Average	84.44%	85.30%	0.86%

The results for Naive Bayes and Term Frequency also indicate the same condition. Stemming in pre-processing task does not help to raise the accuracy. The accuracy difference in this experiment is 2.34%. This achievement is higher than the previous experiments using TF-IDF. More detail about the results using Naive Bayes and Term Frequency is shown in Table 8.

Table 8. Experiments Results Using Naive Bayes and Term Frequency

Number of Datasets	Accuracy		Accuracy Difference
	Term Frequency + Stemming	Term Frequency + Without Stemming	
1500	88.00%	88.67%	0.67%
1750	86.29%	89.14%	2.85%
2000	85.50%	89.00%	3.50%
Average	86.60%	88.94%	2.34%

On the other hand, our experiments using SVM also denote that stemming in pre-processing task does not enhance the accuracy. Table 9 depicts the experiment results using SVM and TF-IDF.

Table 9. Experiments Results Using SVM and TF-IDF

Number of Datasets	Accuracy		Accuracy Difference
	TF-IDF + Stemming	TF-IDF + Without Stemming	
1500	89.56%	91.33%	1.77%
1750	91.62%	92.00%	0.38%
2000	91.50%	91.67%	0.17%
Average	90.89%	91.67%	0.77%

Table 10 shows the results accuracy using SVM and Term Frequency. Based on our experiments, the attainment of accuracy without stemming is better accuracy than when stemming is conducted in pre-processing. According to both experiments using SVM, the accuracy differences are almost the same. The accuracy difference using SVM and TF-IDF obtained 0.77%, whereas for SVM and Term Frequency obtained 0.87%. Based on our experiments, it is clear that pre-processing task without stemming has better accuracy than when stemming is conducted in pre-processing.

Table 10. Experiments Results Using SVM and Term Frequency

Number of Datasets	Accuracy		Accuracy Difference
	TF-IDF + Stemming	TF-IDF + Without Stemming	
1500	88.67%	89.11%	0.44%
1750	91.05%	91.62%	0.57%
2000	90.67%	92.00%	1.33%
Average	90.13%	90.91%	0.78%

4. Conclusion

This paper has examined the stemming influence on tweet classification. To examine the stemming influence, this work has compared between the two approaches in pre-processing task. The first pre-processing steps involved stemming and the other one does not involve stemming. According to the experiments, it is observed that all accuracy results in tweet classification tend to decrease. Moreover, stemming task does not help to raise the accuracy either using SVM or Naive Bayes algorithm. Finally, this work summarized that stemming process does not affect significantly towards the accuracy performance.

References

[1] Basnur PW, Sensuse DI. Pengklasifikasian Otomatis Berbasis Ontologi untuk Artikel Berbahasa Indonesia. *MAKARA of Technology Series*. 2010; 14(1): 29-35.
[2] Ramasubramanian C, Ramya R. Effective Pre-processing Activities in Text Mining Using Improved Porter's Stemming Algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*. 2013; 2(12): 4536-4538.
[3] Sharma D, Jain S. Evaluation of Stemming and Stop Word Techniques on Text Classification Problem. *International Journal of Scientific Research in Computer Science and Engineering*. 2015; 3(2): 1-4.
[4] Gaustad T, Bouma G. Accurate Stemming of Dutch for Text Classification. *Language Computing*. 2002; 45(1): 104-177.
[5] Yu B. An Evaluation of Text Classification Methods for Literary Study. *Literary and Linguistic Computing*. 2008; 23(3): 327-343.
[6] Torunoğlu D, Çakırman E, Ganiz MC, Akyokuş S, Gürbüz MZ. *Analysis of Preprocessing Methods on Classification of Turkish Texts*. In Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on IEEE. 2011: 112-117.
[7] Toman M, Tesar R, Jezek K. *Influence of Word Normalization on Text Classification*. Proceedings of InSciT (2006). 2006; 4: 354-358.
[8] Wahbeh A, Al-Kabi M, Al-Radaideh Q, Al-Shawakfa E, Alsmadi I. The Effect of Stemming on Arabic Text Classifiation:An Empirical Study. *International Journal of Information Retrieval Research*. 2011; 1(3): 54-70.
[9] Hidayatullah AF. *The Influence of Indonesian Stemming on Indonesian Tweet Sentiment Analysis*. Proceeding of International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2015). Palembang, Indonesia. 2015; 2(1): 182-187.
[10] Agusta L. *Perbandingan Algoritma Stemming Porter dengan Algoritma Nazief & Adriani untuk Stemming Dokumen Teks Bahasa Indonesia*. In Proceeding Konferensi Nasional Sistem dan Informatika. Bali, Indonesia. 2009: 196-201.
[11] Asian J, Williams HE, Tahaghoghi SMM. *Stemming Indonesian*. Proceedings of the Twenty-eighth Australasian conference on Computer Science. 2005; 38: 307-314.
[12] Yamamoto M, Church KW. Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in A Corpus. *Computational Linguistics*. 2001; 27(1): 1-30.
[13] Srividhya V, Anitha R. Evaluating Preprocessing Techniques in Text Categorization. *International Journal of Computer Science and Application*. 2010; 47(11): 49-51.
[14] Manning C, Raghavan P, Schutze H. Introduction to Information Retrieval. Cambridge University Press. 2009.