Comparison Nazief Adriani And CS Stemmer Algorithm For Stemm Real Data

Hari Widayanto[#1], Arief Fatchul Huda[#2]

[1]*Telkom University,* [2]*Universitas Islam Negeri Bandung, Indonesia*

[1]hariw73@gmail.com, [2]afhuda@gmail.com

Abstract

Stemming is a processs to find root word from its compounded form by removing all affixes are attached on it. Stemmer was applied in various text mining application to improve application performance, such as in Information Retrieval stemmer could improve performance by providing variant morphological searched terms and reduce size of index [9]. In word based text compression, stemmer could simplify the dictionary as various word from could be represented by one word [6]. Besides reduce size of document index, stemmer could increase text retrieval accuracy [10]. In text classification stemmer reduce the number of features [18].
The first Indonesian stemmer was developed by Nazief-Adriani then Jelita Asian improved the algorithm called confix stripping (CS) stemmer. There were heaps of improvement was done by CS stemmer so it is highest accuracy stemmer algorithm. Experiment would be performed to compare the accuracy between Nazief – Adriani and CS stemmeralgorithm for stemm words were extracted from online news, Republika.

Keywords : Stemming, Indonesian, Nazief-Adriani, CS stemmer

I INTRODUCTION

Stemmer have been applied in text clustering, text classification, summarization, information retrieval, and word-based text compression. In text clustering, stemmer to simplify words without losing meaning [19] so the size of data set will be reduced; Stemmer can reduce number of words in text classification so the number of features will be reduced[18]; in text summarization stemmer can raise the efficiency[20]; In information retrieval, stemmer reduce the number of document index so increase the performance of information retrieval[9], providing variants morphological in searched term so the performance of information retrieval is improved[9], and stemmer is used in information retrieval to increase accuracy[10]. In word-based text compression, stemmer maximize the use of dictionary by keep only basic word in dictionary[6].

II LITERATURE REVIEW

Indonesian Morphology

There are variations of affixes including prefixes, suffixes, infixes, and confixes. Indonesian also has repeated words, combinations of affixes, and combinations of affixes with repeated words. Moreover, Indonesian also has compound words that are written together when attached to a prefix and a suffix. For example the *"pemerintah"* (the government, a government), *"pemerintahan"* (government, government administration), *"diperintah"* (be ruled, be ordered), and *"perintahnya"* (his/her order), which can all be stemmed to *"perintah"* (command, order); and *"buku-buku"* (books), *"bukumu"* (our book), and *"pembukuan"* (accounting), which can all be stemmed to *"buku"* (book). Adding affixes can change the meaning of Indonesian words greatly. These kinds of affixes are called derivative affix.

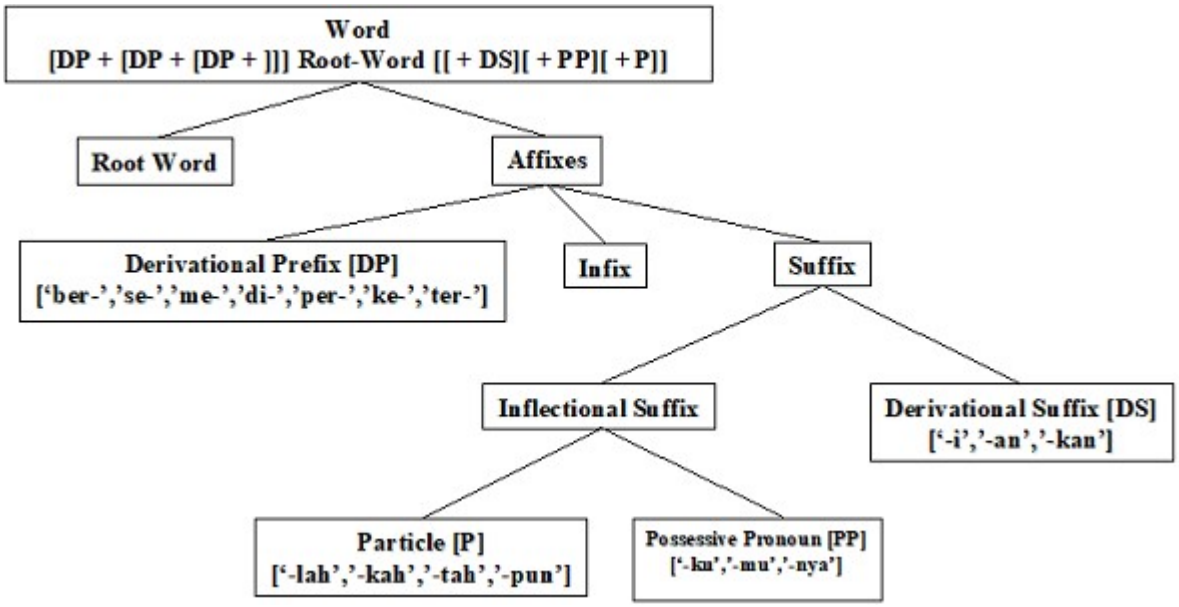

Figure 1. Indonesian Morphology

III RESEARCH METHOD

A. Nazief -Adriani Stemmer Algorithm

The stemming scheme of S_NA is described in an unpublished technical report from the University of Indonesia. [4] The algorithm is based on comprehensive morphological rules that group together and encapsulate allowed and disallowed affixes, including prefixes, suffixes, and confixes (combination of prefixes and suffixes), which are also known as circumfixes. This classification of affixes leads to the rules:

[DP+[DP+[DP+]]] root-word [[+DS][+PP][+P]]

where DP is derivational prefix; DS is derivational suffix; PP is possessive pronoun; and P is particle (both PP and P are inflectional affixes). We apply this order and knowledge of basic rules of the Indonesian language as the foundation of our stemming approach:

1. Words of three or fewer characters cannot contain affixes, so no stemming is performed on such short words.
2. In practice, affixes are never repeated, so a stemmer should remove only one of a set of repeating affixes.
3. We can use confix restriction during stemming to rule out invalid affix combinations. The list of invalid affix combinations are listed in Table 4.
4. If characters are being restored after prefix removal, we perform recoding if necessary. We explain this in Step 5 of the next section..

Detailed algorithm as follows :

1. At the start of processing, and at each step, check the current word against the root word dictionary; if the lookup succeeds, the word is considered to be a stem, and processing stops.
2. Remove inflectional suffixes. As described before, inflectional suffixes do not affect the spelling of the word they attach to, and multiple inflectional suffixes always appear in order. We first remove any inflectional particle (P) suffixes {*"-kah", "-lah", "-tah",* or *"-pun"*}, and then any inflectional possessive pronoun (PP) suffixes {*"-ku", "-mu",* or *"-nya"*}. For example, the word *"bajumulah"* (it is your cloth that) is stemmed first to *"bajumu"* (your cloth), and then to *"baju"* (cloth). This is present in the dictionary, so stemming stops.
   According to our affix model, this leaves the stem with derivational affixes, indicated as:
   [[[DP+]DP+]DP+] root-word [+DS]
3. Remove any derivational suffixes {*"-i", "-kan"*, and *"-an"*}. In our affix model, this leaves:
   [[[DP+]DP+]DP+] root-word
   Consider the word *"membelikan"* (to buy for); this is stemmed to *"membeli"* (to buy). Since this is not a valid dictionary root word, we proceed to prefix removal in the next step.
4. Remove any derivational prefixes {*"be-", "di-", "ke-", "me-", "pe-", "se-", and "te-"*}:
   a) Stop processing if:

- the identified prefix forms an invalid affix pair with a suffix that was removed in Step 3; the invalid pairs are listed in Table 4;
- the identified prefix is identical to a previously removed prefix; or
- three prefixes have already been removed.

b) Identify the prefix type and disambiguate if necessary. Prefixes may be of two types:
plain The prefixes {*"di-", "ke-", "se-"*} can be removed directly.
complex Prefixes starting with {*"be-", "te-", "me-", or "pe-"*} must be further disambiguated using the rules described in Table 1 because these have different variants. The prefix *"me-"* could become *"mem-"*, *"men-"*, *"meny-"*, or *"meng-"* depending on the letters at the beginning of the root word.
In the previous step, we partially stemmed the word "membelikan" to "membeli". We now remove the prefix *"mem-"* to obtain *"beli"*. This is a valid root, and so processing stops. For the word *"mempertinggi"* (to heighten), we remove the prefix *"mem-"* to obtain the word *"pertinggi"* (to heighten). If none of the prefixes above match, processing stops, and the root word was not found.

Rule	Construct	Return
1	berV. . .	ber-V. . . | be-rV. . .
2	berCAP. . .	ber-CAP . . . where C!='r' and P!='er'
3	berCAerV. . .	ber-CaerV . . . where C!='r'
4	belajar. . .	bel-ajar. . .
5	beC1erC2. . .	be-C1erC2 . . .where C1!={'r'| 'l'}
6	terV . . .	ter-V . . . | te-rV . . .
7	terCerV . . .	ter-CerV . . . where C!='r'
8	terCP . . .	ter-CP. . . where C!='r' and P!='er'
9	teC1erC2. . .	te-C1erC2. . .where C1!='r'
10	me{l|r|w|y}V. . .	me-{l|r|w|y}V. . .
11	mem{b|f|v}. . .	mem-{b|f|v}. . .
12	mempe{r|l}. . .	mem-pe. . .
13	mem{rV|V}. . .	me-m{rV|V}. . . | me-p{rV|V}. . .
14	men{c|d|j|z}. . .	men-{c|d|j|z}. . .
15	menV. . .	me-nV. . . | me-tV. . .
16	meng{g|h|q}. . .	meng-{g|h|q}. . .
17	mengV. . .	meng-V. . . | meng-kV. . .
18	menyV. . .	meny-sV. . .
19	mempV. . .	mem-pV. . .where V!='e'
20	pe{w|y}V. . .	pe-{w|y}V. . .
21	perV. . .	per-V. . . | pe-rV. . .
22	perCAP. . .	per-CAP. . .where C!='r' and P!='er'
23	perCAerV. . .	per-CaerV. . .where C!='r'
24	pem{b|f|v}. . .	pem-{b|f|v}. . .
25	pem{rV|V}. . .	pe-m{rV|V}. . . | pe-p{rV|V}. . .
26	pen{c|d|j|z}. . .	pen-{c|d|j|z}. . .
27	penV. . .	pe-nV. . . | pe-tV. . .
28	peng{g|h|q}. . .	peng-{g|h|q}. . .
29	pengV. . .	peng-V. . . | peng-kV. . .
30	penyV. . .	peny-sV. . .
31	pelV. . .	pe-lV. . . Exception: for "pelajar", return ajar
32	peCerV. . .	per-erV. . .where C!={r|w|y|l|m|n}
33	peCP. . .	pe-CP. . .where C!={r|w|y|l|m|n} and P!='er'

Table1 1 : Template formulas for derivation prefix rules.

c) If a dictionary lookup for the current word fails, we repeat Step 4 (this is a recursive process). If the word is found in the dictionary, processing stops. After the recursive prefix removal, the word *"pertinggi"* becomes the correct stem *"tinggi"* (high) that is found in the dictionary after removal of the prefix *"per-"*. If the word is not found after recursive prefix removal and the three conditions in 4a are not violated yet, proceed to the next step. For example, the word *"menangkap"* (to catch) satisfies Rule 15 for the prefix *"me-"* (the initial prefix *"men-"* is followed by a vowel *"a-"*). After removing *"men-"*, we obtain *"angkap"*, which is not a valid root word. Further recursive prefix removal does not succeed since there is no other valid prefix to be removed.

5. If, after recursive prefix removal, the word has still not been found, we check whether recoding is possible by examining the last column of Table 3.1, which shows the prefix variants and recoding characters to use when the root word starts with a certain letter, or when the first syllable of the root word ends with a certain letter or fragment. A recoding character is a lowercase letter following the hyphen and outside the braces. Not all prefixes have a recoding character.
   Fromthe example *"menangkap"* above, there are two possible recoding characters based on Rule 15, *"n"* (as in *"men-nV. . . "*) and *"t"* (as in *"men-tV. . . "*). This is somewhat exceptional; inmost cases there is only one recoding character. The algorithmprepends *"n"* to *"angkap"* to obtain *"nangkap"*, and returns to Step 4. Since this is not a valid root word, *"t"* is prepended instead to obtain *"tangkap"* (catch), and we return to Step 4. Since *"tangkap"* is a valid root word, processing stops.
6. If all steps are unsuccessful, the algorithm returns the original unstemmed word. Although.
   Although the confixes are not explicitly removed in the above steps, they are indirectly removed by the removal of prefixes and suffixes. There may be some exception cases. For example, the confix *"pe-an"* in the word *"pengusutan"* could mean either *"entanglement"*, which is derived from *"kusut"* (tangled) or *"examination, investigation"*, which is derived from *"usut"* (examine). Without using context, neither an automatic stemmer or humans can tell which is the correct stem.

B. Confix Stripping Stemmer Algorithm

To address the limitations of the S_NA algorithm, CS Stemmer propose the following improvements:

1. Using a more complete dictionary
2. Adding rules to deal with plurals—when plurals, such as *"buku-buku"* (books) are encountered, we propose stemming these to *"buku"* (book). However, care must be taken with other hyphenated words such as *"bolak-balik"* (to and fro), *"berbalas-balasan"* (mutual action or interaction) and *"seolah-olah"* (as though). For these later examples, we propose stemming the words preceding and following the hyphen separately and then, if the words have the same root word, to return the singular form. For example, in the case of *"berbalas-balasan"*, both *"berbalas"* and *"balasan"* stem to *"balas"* (response or answer), and this is returned. In contrast, the words *"bolak"* and *"balik"* do not have the same stem, and so *"bolak-balik"* is returned as the stem; in this case, this is the correct action, and the approach works for many hyphenated non-plurals.
3. Adding prefixes and suffixes, and additional rules:
   a. Adding the particle (inflection suffix) *"-pun"* to the list of suffixes to be stemmed. This is used in words such as *"siapapun"* (where the root word is *"siapa"* (who?). The particle "-pun" is not supposed to be attached to the root word except for conjunction; however, from observation, people often attach this particle with the root word. Therefore, we choose to deal with this common mistake.
   b. For the prefix type *"te-"*, we have added a new condition (Rule 35) in Table 6 Previously, words such as *"terpercaya"* (the most trusted) are not stemmed. By the addition of this new rule, it is correctly stemmed as *"percaya"* (believe).

New Rule	Construct	Return
34	terC1erC2. . .	ter-C1erC2. . .where C1!='r'
35	peC1erC2. . .	pe-C1erC2. . .where C1!={r|w|y|l|m|n}

Modified Rule	Construct	Return
12	mempe ...	mem-pe ...
16	meng{g|h|q|k}. . .	meng-{g|h|q|k}. . .

Table2 : Additional and modified template formulas for derivation prefix rules in Table 1

   c. For the prefix type *"pe-"*, we have added Rule 36 in Table 6 so that words such as *"pekerja"* (worker) and *"peserta"* (member) are stemmed correctly to *"kerja"* (to work) and *"serta"* (along with, as well as), rather than not stemmed as the prefix is not recognised by S_NA algorithm.
   d. For the prefix type *"me-",* we have modified Rule 12 so that words such as *"mempengaruhi"* (to influence) can be stemmed correctly *"pengaruh"* (influence) instead of not successfully stemmed.
   e. We havemodified Rule 16 for the prefix type "me-", so that the word "mengkritik" (to criticise) can be stemmed correctly to *"kritik"* (critics).
4. Adjusting rule precedence. The order in which rules are applied affects the outcome of the stemming operation. Consider an example where inflectional suffix removal fails. The word *"bertingkah"* (to

behave) is formed from the prefix *"ber-"* and the root word *"tingkah"* (behaviour). However, the algorithm will remove the suffix *"-kah"* to obtain the word *"berting"*, and then remove the prefix *"ber-"* to obtain the valid word *"ting"* (lamp). This particular problem arises only in limited cases with specific prefixes and particles. The list of rules that have been adjusted are:

a. If a word is prefixed with "be-" and suffixed with the inflection suffix "-lah", try to remove prefix before the suffix. This addresses problems with words such as "bermasalah" (having a problem) and *"bersekolah"* (be at school) to be stemmed correctly to *"masalah"* (problem) and *"sekolah"* (school) instead of the erroneous *"seko"* (spy) and *"masa"* (time, period).
b. If a word is prefixed with *"be-"* and suffixed with the derivation suffix *"-an"*, try to remove prefix before the suffix. This solves problems with, for example, *"bertahan"* (to hold out) is stemmed correctly to *"tahan"* (to last, to hold out) instead of *"tah"* which is a shortened form of *"entah"* (who knows).
c. If a word is prefixed with *"me-"* and suffixed with the derivation suffix *"-i"*, try to remove the prefix before the suffix. This solves problems with, for example, *"mencapai"* (to reach) stemmed correctly to *"capai"* (to reach) instead of *"capa"* (game of heads or tails).
d. If a word is prefixed with *"di-"* and suffixed with the derivation suffix *"-i"*, try to remove the prefix before the suffix. This solves problems with, for example, *"dimulai"* (to be started) stemmed to *"mulai"* (to start) rather than *"mula"* (beginning).
e. If a word is prefixed with *"pe-"* and suffixed with the derivation suffix *"-i"*, try to remove prefix before the suffix. This solves problems with, for example, *"petani"* (farmer) stemmed to *"tani"* (farm, farmer) rather than *"petan"* (a game of hide and seek).

If a word is prefixed with *"ter-"* and suffixed with the derivation suffix *"-i"*, try to remove prefix before the suffix. This solves problems with, for example, *"terabai"* (ignored) stemmed to *"abai"* (neglectful) instead of *"aba"* (father).

C. Republika Online News

Contents of news was grabed from index page of Republika Online News website (http://www.republika.co.id/index/). News were showed based on date published splitted in several pages, each page contains 20 news. Text news content was placed between tag " and " ".

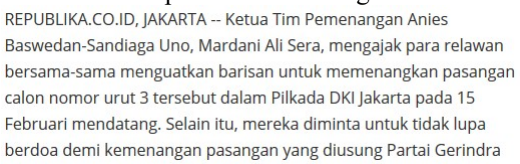
REPUBLIKA.CO.ID, JAKARTA -- Ketua Tim Pemenangan Anies Baswedan-Sandiaga Uno, Mardani Ali Sera, mengajak para relawan bersama-sama menguatkan barisan untuk memenangkan pasangan calon nomor urut 3 tersebut dalam Pilkada DKI Jakarta pada 15 Februari mendatang. Selain itu, mereka diminta untuk tidak lupa berdoa demi kemenangan pasangan yang diusung Partai Gerindra

Figure 2 : Republika Online News

```
<div class="content-detail">

    <p>REPUBLIKA.CO.ID, JAKARTA -- Ketua Tim Pemenangan Anies Baswedan-Sandiaga Uno, Mardani Ali Sera, mengajak para relawan bersama-sama menguatkan barisan untuk memenangkan pasangan calon nomor urut 3 tersebut dalam Pilkada DKI Jakarta pada 15 Februari mendatang. Selain itu, mereka diminta untuk tidak lupa berdoa demi kemenangan pasangan yang diusung Partai Gerindra dan PKS ini.</p>
```

Figure 3 : Republika Online News in Html Format

From news was taken for two months, such from February,1 2017 to Maret, 31 2017 were obtained 26.462 news consists of 2.141.421 words, from those was got 79.453 unique words.

IV RESULT AND DISCUSSION

A. Result

Table2 show performance comparison of stemming process of stemmer algorithm Nazief - Adriani Stemmer (S_NA) and Confix Stripping Stemmer (CS) algorithm for stemm words that extracted from online news Republika. In this table shown that CS stemmer has higher performance in accuracy than S_NA stemmer, but CS stemmer has more time consumed.

Both stemmers have low accuracy, this is not as high as the performance achieved by CS stemmer which reached 97% in previous research. This is because the previous research did not use all the words in the text while in this research using all the words in the news

Stemmer	Root Word	Success	Failed	Total Result	Ach (%)	Duration (s)
S_NA	5.114	26.181	53.272	79.453	**32,95**	**4.873,55**
CS	5.421	29.094	50.359	79.453	**36,62**	**4.980,62**

Table 2 Stemming Result

B. Cause Failures

Failed stemming causes categorization for Republika *were* shown in table 3. Failed was caused by foreing language by 19.453 words follwed by name by 17.196 words, misspelling by 10.506 words, unidentified affix by 1.038 words . Failed caused by rule was rule 18.

Category	Count
foreign	19.453
name	17.196
misspelling	10.506
unidentified affix	1.038
dictionary	240
composite	209
True Negatif	98
rule18	10
not word	2
Total Result	**48.752**

Table 3 Category of Failed Stemmed for Republika

C. Result Comparison S_NA vs CS Stemmer

CS Stemmer has better result than S_NA Stemmer in, but CS Stemmer has more time consumed compared with S_NA stemmer. Stemming process results comparison between S_NA and CS Stemmer (CS) for each data source is shown in table 4-5, 4-6, and 4-7. Number in vertical box are features belong to S_NA stemmer and number in horizontal box are features belong to CS stemmer.

	CS				
S_NA	RootWord	Stemmed	UnKnown	UnStemmed	**Total Result**
RootWord	5.114				**5.114**
Stemmed		20.875		174	**21.049**
UnKnown		87	32.816	514	**33.417**
UnStemmed	319	2.757	213	16.584	**19.873**
Total Result	**5.433**	**23.719**	**33.029**	**17.272**	**79.453**

Table Error! No text of specified style in document. Stemming Result Comparison S_NA vs CS for Data of Republika

There were improvement by CS from S_NA :

- words stemmed as UnStemmed by S_NA but stemmed correctly by CS. The number were 319. Those were impact of addition rule by CS : adding rule to deal with plural, example : *"perlahan-lahan"*.

- words stemmed as UnKnown by S_NA but stemmed correctly by CS. The number were 87. Those were impact of adding rule to deal with plural and modify prefix rule number 16, example : *"mengklaim"*
- words stemmed as UnStemmed by S_NA but stemmed correctly by CS. The number were 2.757 words. Those were impact of adding rule to deal with plural and rule precedence, example : *"menjadi"*

V CONCLUSION

There were big improvement from S_NA to CS stemmer algorithm by adding several prefix rules and modify prefix rules particularly adding rule presedence. The most influencial of improvement by CS Stemmer (CS) was adding rule deal with plural as sign as hiphenation "-" in the words follwed by rule precedence.

REFERENCE

[1] F. Tala. "A study of stemming effects on information retrieval in Bahasa Indonesia". *Master's thesis, University of Amsterdam, Netherlands*, 2003.

[2] Andrea H. "Laskar Pelangi". *Bentang Pustaka,* 2005.

[3] Salim B. "Tarjamah Riyadhus Shalihin". *PT. Al Ma'arif, Bandung,* 1995.

[4] Jelita Asian. "Effective Techniques for Indonesian Text Retrieval". *School of Computer Science and Information Technology, Science, Engineering, and Technology Portfolio, RMIT University, Melbourne, Victoria, Australia*, 2007

[5] A.Z. Arifin, I.P.A.K. Mahendra, and H.T. Ciptaningtyas, "Enhanced confix stripping stemmer and ants algorithm for classifying news document in Indonesian language*," in International Conference on Information & Communication Technology and Systems*, 2009, pp. 149-158

[6] A. Sinaga, Adiwijaya, and H. Nugroho. "Development of word-based text compression algorithm for Indonesian language document". *3rd International Conference on Information and Communication Technology (ICOICT)*, 2015, pp. 450-454, DOI: 10.1109/ICoICT.2015.7231466, Bali.

[7] F. Ahmad, M. Yusoff, and T. M. T. Sembok. "Experiments with a Stemming Algorithm for Malay Words". *Journal of the American Society for Information Science*, 47(12):909–918, 1996

[8] M. Adriani, J. Asian, B. Nazief and et al., "Stemming Indonesian: a confix-stripping approach," *in ACM Transactions on Asian Language Information Processsing*, vol. 6, 2007, pp. 1-33.

[9] W. Frakes. "Stemming algorithms." *In Information Retrieval: Data Structures and Algorithms*, pages 131–160. *Prentice Hall, Englewood Cliffs, New Jersey*, 1992

[10] D. Sharma, M. Cse, "Stemming algorithms: a comparative study and their analysis," *in International Journal of Applied Information Systems (IJAIS)*, vol. 4, no. 3, pp. 7-12, 2012

[11] R. Setiawan, A. Kurniawan, W. Budiharto, I. H. Kartowisastro, H. Prabowo, "Flexible Affix Classification for Stemming Indonesian Language" *Doctor of Computer Science Department Bina Nusantara University Jakarta, Indonesia*, 2016

[12] "Kamus Besar Bahasa Indonesia," 4th ed., *Departemen Pendidikan Nasional*, 2008.

[13] H. Alwi, S. Dardjowidjojo, H. Lapoliwa, A. M. Moeliono, "Tata Bahasa Baku Bahasa Indonesia," *3rd ed., Balai Pustaka*, 2003.

[14] A. Z. Arifin and A. N. Setiono. "Classification of event news documents in Indonesian language using single pass clustering algorithm." *In Proceedings of the Seminar on Intelligent Tech- nology and Its Applications (SITIA), Surabaya, Indonesia*, 2002.

[15] M.F. Porter. "An algorithm for suffix stripping. " *In Program,* volume 14, pages 130–137, 1980.

[16] Suhartono, Derwin and D. Christiandy. “Lemmatization Technique in Bahasa : Indonesian Language” *Computer Science Department, Bina Nusantara University, Jakarta, Indonesia*. 2014

[17] V.B. Vega. “Indexing the Indonesian Web : Language Identification and Miscellaneous Issues” *Department of Computer Science National University of Singapore, Presented at Tenth International World Wide Web Conference, Hong Kong,* 2001

[18] T. Gaustad and G. Bouma. “Accurate Stemming of Dutch for Text Classification” *Language and Computers,* 2002

[19] A. Khare and A. N. Jadhav. “An Efficient Concept-Based Mining Model For Enhancing Text Clustering”. *International Journal of Advanced Engineering Technology.* 2011

[20] V. Gupta and G. S. Lehal. “Punjabi Language Stemmer for nouns and proper names Vishal” *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), IJCNLP* 2011, pages 35–39, *Chiang Mai, Thailand*, November 8, 2011