International Journal on Advanced Science Engineering Information Technology

Vol.7 (2017) No. 5
ISSN: 2088-5334

Improving Stemming Algorithm Using Morphological Rules

Titin Winarti[#], DJati Kerami[*], Lussiana ETP[+], Sunny Arief Sudiro[+]

[#]*Department of Informatics, Semarang University, Soekarno Hatta Street, Semarang, 50196, Indonesia*
E-mail: titin@usm.ac.id

[*]*Department of Mathematics and Natural Sciences, University of Indonesia, Pondok Cina, Depok, 16424, Indonesia*
E-mail: djatikr@sci.ui.ac.id

[+]*Department of Computer System, School of Information Management Jakarta, Radio Dalam Street, 12140, Indonesia*
E-mail: {lussiana,sunny}@jak-stik.ac.id

***Abstract*— Stemming words to remove suffixes has applications in text search, translation machine, summarization document, and text classification. For example, Indonesian stemming reduces the words "kebaikan", "perbaikan", "memperbaiki" and "sebaik-baiknya" to their common morphological root "baik". In text search, this permits a search for a player to find documents containing all words with the stem play. In the Indonesian language, stemming is of crucial importance: words have prefixes, suffixes, infixes, and confixes that make them match to relate difficult words. This research proposed a stemmer with more accurate word results by employing an algorithm which gave more than one word candidate results and more than one affix combinations. New stemming algorithm is called CAT stemming algorithm. Here, the word results did not depend on the order of the morphological rule. All rules were checked, and the word results were kept in a candidate list. To make an efficient stemmer, two kinds of word lists (vocabularies) were used: words that had more than one candidate words and list of root word as a candidate reference. The final word results were selected with several rules. This strategy was proved to have a better result than the two most known about Indonesian stemmers. The experiments showed that the proposed approach gave higher accuracy than the compared systems known.**

Keywords— **stemming; information retrieval; morphological rule**

I. INTRODUCTION

Stemming is a core natural language processing technique for efficient and effective Information Retrieval [1], and one that is widely accepted by users. It is used to transform word variants to their common root of the word by applying it in most cases of morphological rules [2]. For example, in text searching, it should permit a user searching by using the query term stemming to find documents that contain the terms stemmer and stems because all share the common root word stem. It also has applications in translation machine [4], document summarization [9], and text classification [8].

For English, stemming is well-understood, with techniques such as those of Lovin and Porter [10] in widespread use. However, stemming for other languages is less well-known: while there are several approaches available for languages such as French [11], Malaysian [7], and Indonesian [5].

Several techniques have been proposed for stemming Indonesian. We evaluate these techniques through a user study, where we compare the performance of the scheme to the results of manual stemming by four native speakers. Our results show that an existing technique, proposed by Nazief and Adriani (1996) in an unpublished technical report, correctly stems around 93% of all word occurrences (or 92% of unique words). After classifying the failure cases, and adding our own rules to address these limitations, we show this can be improved to 95% for both unique and all word occurrences. We believe that adding a complete dictionary of root words would improve these results even further. We conclude that our modified Nazief and Adriani stemmer should be used in practice for stemming Indonesian.

All of the previous works used a way in solving morphological problems where there was only one rule path that could be executed for a single input. The formula was used to check particle, suffix, and prefix in rule order [6]. This was not suitable for an ambiguous morphological word such as "mereka", "menggulai" or "kemeja", where there should be more than one correct result, depending on the

context. This paper proposes an algorithm where it can have more than one candidate for those ambiguous words.

The rule order of prefix and suffix gives another error of stemmed word [12]. The error is obtained when the input words end in lexical similar with suffixes/ particle/ possessive pronoun. For example, the word "penemu" is stemmed into "pene"+"mu" because the possessive pronoun rule is first executed before the prefix rule. Although Adriani [5] has tried to fix the problem by adding several exception rules to check the prefix first before the suffix, this approach still can't handle the exception conditions if the conditions are not listed in the exception rules. In Adriani [5], the exception rules are "ber"+word+"lah" (such as word "bersekolah"), "ber" + word +"an" (such as word "berlainan"), "di"-"i", "ter"-"i", "me"-"i" and "pe"-"i". It still can't handle other patterns such as words "penemu", "penanya", etc. The algorithm proposed in this paper will give all candidates and then select the best candidate.

Another weakness of Adriani [5], there is no dictionary of stemmed word list involved which causes inaccurate result such as "perbaikan" is stemmed into "bai" root word with "per" as the prefix and "kan" as the suffix. This is due to the rule position of "kan" is prior to the rule position of "an". One can argue that the solution is to change the rule order where rule of "an" affix is positioned prior than the "kan" affix rule, but then this solution will result in similar problems for words with "kan" affix such as "menarikan" which then will be morphologically analysed into "me", "tarik" and "an" [12].

Meanwhile, in Adriani [5], a complete root of word dictionary is used to handle this problem. For the word "perbaikan", because the root of word "bai" is not in the dictionary then the system will not choose "per" – "bai" – "kan". Instead, the stemmer will choose "per" - "baik" - "an", due to the root of word "baik" is in the dictionary.

Although Adriani [5] stemmer gives a better result, it consumes much time because, for each rule, the resulted word is searched into the root of the word in the dictionary. As an alternative solution, this paper proposes the usage of a restricted list of the root of a word called as vocabulary which size is less than the complete dictionary used by Adriani [5] stemmer. By using this, the proposed stemmer will give good accuracy score such as Adriani [5] stemmer with less complexity.

II. Material and Methods

A. Nazief Adriani Algorithm

The stemming scheme of Nazief and Adriani is described in an unpublished technical report from the University of Indonesia (1996). In this section, we describe the steps of the algorithm, and illustrate each with examples; however, for compactness, we omit the detail of selected rule tables. We refer to this approach like Nazief.

The algorithm is based on comprehensive morphological rules that group together and encapsulate allowed and disallowed affixes, including prefixes, suffixes, infixes (insertions) and confixes (combination of prefixes and suffixes). The algorithm also supports recoding, an approach to restore an initial letter that was removed from the root of the word prior to prepending prefix. In addition, the algorithm makes use of an auxiliary dictionary of the root of words that are used in most steps to check if the stemming has arrived at a root of the word.

Before considering how the scheme works, we consider the basic groupings of affixes used as a basis for the approach, and how these definitions are combined to form a framework to implement the rules. The scheme groups affixes into the following categories:

1) Inflectional Suffixes: the set of suffixes that do not alter the root of the word. For example, "pulang" (sit) may be suffixed with "-lah" to give "pulanglah" (please sit). The inflections are further divided into:

- Particles (P): including "-lah" and "-kah", as used in words such as "duduklah" (please sit).
- Possessive pronouns (PP): including "-ku", "-mu", and "-nya", as used in "ibunya" (a third person possessive form of "mother"). Particle and possessive pronoun inflections can appear together and, if they do, possessive pronouns appear before particles. A word can have at most one particle and one possessive pronoun, and these may be applied directly to the root of words or to words that have a derivation suffix. For example, "makan" (to eat) may be appended with derivation suffix "-an" to give "makanan" (food). This can be suffixed with "-nya" to give "makanannya" (a possessive form of "food")

2) Derivational Suffixes: the set of suffixes that are directly applied to the root of words. There can be only one derivation suffix per word. For example, the word "lapor" (to report) can be suffixed by the derivation suffix "–kan" to become "laporkan" (go to report). In turn, this can be suffixed with, for example, an inflectional suffix "-lah" to become "laporkanlah" (please go to report).

3) Derivational Prefixes: the set of prefixes that are applied either directly to the root of words, or to words that have up to two other derivational prefixes. For example, the derivational prefixes "mem-" and "per-"may be prepended to "indahkannya" to give "memperindahkannya" (the act of beautifying).

The classification of affixes as inflections and derivations leads to an order of use:

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

TABLE I
DISALLOWED PREFIX AND SUFFIX COMBINATIONS [5]

Prefix	Disallowed suffixes	Examples
be-	-i	be-kerja-i → be-kerja
di-	-an	di-jual-an → di-jual-kan
ke-	-i, -kan	ke-sakit-kan → ke-sakit-an ke-sakit-i → ter-sakit-i
me-	-an	me-latih-an → me-latih-kan
se-	-i, -kan	Se-nasib-kan → se-nasib
te-	-an	ter-bawa-an→ ter-bawa-kan

Table 1 is disallowing prefix and suffix combinations. The only exception is that the root of word "tahu" is permitted with the prefix "ke-" and the suffix "-i".

The square brackets indicate that an affix is optional. The previous definition forms the basis of the rules used in the approach. However, there are exceptions and limitations that are incorporated in the rules:

1) Not All Combinations are Possible: after a word is prefixed with "di-", the word is not allowed to be suffixed with "-an". A complete list is shown in Table 1.

2) The Same Affix Cannot be Repeatedly Applied: after a word is prefixed with "te-" or one of its variations, it is not possible to repeat the prefix "te-" or any of those variations.

3) If a Word Has One or Two Characters: then stemming is not attempted.

4) Adding a Prefix May Change the Root of Word or a Previously Applied Prefix: we discuss this further in our description of the rules. To illustrate, consider "meng-" that has the variations "mem-", "meng-","meny-", and "men-". Some of these may change the prefix of a word, for example, for the root of word "sapu" (broom), the variation applied is "meny-" to produce the word "menyapu" (to sweep) in which the "s" is removed.

The latter complication requires that an effective Indonesian stemming algorithm is able to add deleted letters through the recoding process. The algorithm itself employs three components: the affix groupings, the order of using rules (and their exceptions), and dictionary. The dictionary is checked after any stemming rule succeeds: if the resultant word is found in the dictionary, then stemming has succeeded in finding a root of the word, the algorithm returns the dictionary word, and then stops; we omit this lookup from each step in our listing rule. In addition, each step checks if the resultant word is less than two characters in length and, if so, no further stemming is attempted.

For each word to be stemmed, the following steps are followed:

1) The unstemmed word is searched in the dictionary. If it is found in the dictionary, it is assumed that the word is a root of the word, and so the word is returned, and the algorithm stops.

2) Inflectional suffixes ("-lah", "-kah", "-ku", "-mu", or "-nya") are removed. If this succeeds and the suffix is a particle ("-lah" or "-kah"), this step is again attempted to remove any inflectional possessive pronoun suffixes ("-ku", "-mu", or "-nya").

3) Derivational suffix ("-i" or "-an") removal is attempted. If this succeeds, Step 4 is attempted. If Step 4 does not succeed:

- If "-an" was removed, and the final letter of the word is "-k", then the "-k" is also removed and Step 4 is reattempted. If that fails, Step 3b is performed. Table 2 is determining the prefix type for words prefixed with "te–". If the prefix "te-" does not match one of the rules in the table, then "none" is returned. Similar rules are used for "be–", "me-", and "pe-".
- The removed suffix ("-i", "-an", or "-kan") is restored.

4) Derivational *Prefix Removal is Attempted:* This has several sub-steps:

- If a suffix is removed in Step 3, then disallowed prefix suffix combinations are checked using the list in Table 1. If a match is found, then the algorithm returns.
- If the current prefix matches any previous prefix, then the algorithm returns.
- If three prefixes have previously been removed, the algorithm returns.
- The prefix type is determined by one of the following steps:
  - If the prefix of the word is "di-", "ke-", or "se-", then the prefix type is "di", "ke", or "se" respectively.
  - If the prefix is "te-", "be-", "me-", or "pe-", then an additional process of extracting character sets to determine the prefix type is required. As an example, the rules for the prefix "te-" are shown in Table 2. Supposed the word be-ing stemmed is "terlambat" (late). After removing "te-" to give "-rlambat", the first set of characters is extracted from the prefix according to the "Set 1" rules. In this case, the letter following the prefix "te-" is "r", and this matches the first five rows of the table. Following "-r-" is "-l-" (Set 2), and so is the third to fifth rows match. Following "-l-" is "-ambat", eliminating the third and fourth rows for Set 3 and determining that the prefix type is "ter-" as shown in the rightmost column.
  - If the first two characters do not match "di-", "ke-", "se-", "te-", "be-", "me-", or "pe-" then the algorithm returns.
- If the prefix type is "none", then the algorithm returns. If the prefix type is not "any", then the prefix type is found in Table 3, the prefix to be removed is found, and the prefix is removed from the word; for compactness, Table 3 shows only the simple cases and those matching with Table 2.
- If the root of the word has not been found, Step 4 is recursively attempted for further prefix removal. If a root of the word is found, the algorithm returns.

TABLE II
DETERMINING THE PREFIX TYPE FOR WORDS [5]

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

- Recoding is performed. This step depends on the prefix type and can result in different prefixes being prepended to the stemmed word and checked in the dictionary. For compactness, we consider only the case of the prefix type "ter-luluh" is shown.
- Table 3 is determining the prefix from the prefix type. Only simple entries and those for the te- prefix type are shown in Tables 2 and 3. In this case, after removing "ter-", an "r-" is prepended to the word. If this new word is not in the dictionary, Step 4 is repeated for the new word. If a root of the word is not found, then "r-" is removed and "ter-" restored, the prefix is set to "none", and the algorithm returns.

TABLE III
DETERMING PREFIX FROM PREFIX TYPE[5]

Prefix type	Prefix to be removed	Examples
di	di-	di-asuh→ asuh
ke	ke-	ke-atas → atas
se	se	se-ekor→ ekor
ter	ter-	ter-buka → buka

1) Having Completed All Steps Unsuccessfully: the algorithm returns to the original word.

B. Improving Nazief Algorithm

In this section, let see Table 4 about affix removal base on the rules of morphology Indonesian.

TABLE IV
THE RULES OF AFFIX REMOVAL

Rule	Affix	Changes Affixes	Examples
1	Ber**V**...	Ber**V**...|be-r**V**...	beroda →be-roda
2	Ber**CAP**...	Ber-**CAP**...where *C*!='r' and **P**!='er'	berkuda→ ber-kuda
3	Ber**CA**er**V**...	Ber-**CA**er**V**...where **C!**='r'	berkerja→ ber-kerja
4	Belajar...	Bel-ajar...	bel-ajar
5	BeC_1erC_2	Be-C_1erC_2...where C_1!={'r'|'l'}	bekerja→ be-kerja
6	Ter**V**...	Ter-**V**...|te-r**V**...	teratas→ter-atas
7	Ter**C**er**V**	Ter-**C**er**V**...where **C**!='r'	tercemar→ ter-cemar
8	Ter**CP**...	Ter-**CP**...where C!='r'and **P**!='er'	terjatuh→ter-jatuh
9	TeC_1erC_2...	Te-C_1erC_2...where C_1!='r'	terencana→ te-rencana
10	Me{l|r|w|y}**V**...	Me-{1|r|w|y}**V**...	merawat→me-rawat
11	Mem{b|f|v}...	Mem-{b|f|v}...	membawa→mem-bawa
12	Mempe	Mem-pe...	mempertaruhkan→ mem-per-**taruh**-kan
13	Mem{r**V**|**V**}...	Me-m{r**V**|**V**}...| Me-p{r**V**|**V**}...	memasak → me-masak
14	Men{c|d|j|z|s}...	Men-{c|d|j|z|s}...	mencuci→men-cuci
15	MenV...	Me-nV...| me-tV...	menari→ me-tari
16	Meng{g|h|q|k}...	Meng-{{g|h|q|k}...	menghadiri→ meng-hadir-i
17	Meng**V**...	Meng-**V**...| meng-kV...|	mengambil→meng-ambil
		(mengV-...if V="e")	mengelak→meng-elak
18	Meny**V**...	Meny-s**V**...	menyapu→ meny-sapu
19	Memp**A**...	Mem-p**A**...where A!='e'	mempertaruhkan→ mem-per-taruh-kan
20	Pe{w|y}**V**...	Pe-{w|y}**V**...	penyapu→peny-sapu
21	Per**V**...	Per-**V**...| pe-r**V**...	peramal→pe-ramal
22	Per**CAP**...	Per-**CAP**... where **C**!='r' and P!='er'	perkataan→per-kata-an
23	Per**CA**er**V**...	Per-**CA**erV... where **C**!='r'	pekerjaan→pe-kerja-an
24	Pem{b|f|v}...	Pem-{b|f|v}...	pembeli→pem-beli
25	Pem{r**V**|**V**}...	Pem{r**V**|**V**}... | Pe-p{r**V**|**V**}...	pembunuh→ pem-bunuh
26	Pen{c|d|j|z}...	Pen-{c|d|j|z}...	pencuri→pen-curi
27	Pen**V**...	Pe-n**V**... | pe-t**V**...	penari→ pe-tari
28	Peng**C**	Peng-**C**	pengkaji→peng-kaji
29	Peng**V**...	Peng-**V**... | peng-k**V**... | (peng**V**-... if **V**="e")	pengukur→peng-ukur
30	Peny**V**...	Peny-s**V**...	penyapu→ peny-sapu
31	Pel**V**...	Pel**V**... kecuali pada kata 'pelajar'	pelajar→pel-ajar
32	Pe**C**er**V**...	Per-er**V**... where C!={r|w|y|l|m|n}	pekerja→pe-kerja
33	Pe**CP**...	Pe-**CP**... where **C**!={r|w|y|l|m|n} and **P**!='er'	pelari→ pe-lari
34	terC_1erC_2...	Ter-C_1erC_2... where C_1!=,,r"	terkejar→ ter-kejar
35	peC_1erC_2...	Pe-C_1erC_2... where C_1!={r|w|y|l|m|n}	pekerja→ pe-kerja

where :
C : consonan, A : vocal or consonan
V : vocal, P : partikel or fragmen

TABLE V
MODIFICATION OF RULES AFFIX REMOVAL

Rule	Affix	Changes Affixes	Examples
11	Mem{b|f|v|p}	Mem-{b|f|v|p}	Membeli→mem-beli
13	Mem{rV|V}..	Me-p{rV|V}...	Memproduksi→ mem-produksi
14	Men{c|d|j|z|s|t}...	Men-{c|d|j|z|s|t}...	Mencari→ men-cari
30	Peng{a|i|u|o}...	Peng-{a|i|u|o}...	Pengukur→peng-ukur
38	Me-mV/C...	Mem-V/C	Memulai→ me-mulai
39	PemV...	Pem-p-V...	Pemasok→ pem-pasok
40	Pe{c|t|s|z}	Pe-{c|t|s|z}...	Penyabar→ pe-sabar

We discuss the reasons why the nazief scheme works well, and what aspects that can be improved. We present a detailed analysis of the failure cases, and propose solutions to these problems. We then present the results, including

the improvements, and describe our modified nazief approach.

The performance of nazief approach is perhaps unsurprising: it is by far the most complex approach, being based closely on the detailed morphological rules of the Indonesian language. In addition, it supports dictionary lookup and progressive stemming, allowing it to evaluate each step to test if a root of the word has been found and to recover from errors by restoring affixes to attempt different combinations. However, despite these features, the algorithm can still be improved.

In summary, three opportunities exist to improve stemming with nazief. First, a more complete and accurate root of word dictionary may reduce errors. Second, features can be added to support stemming of hyphenated words. Last, new rules and adjustments to rule precedence may reduce over and under stemming, as well as support affixes not currently catered for in the algorithm. We will discuss the improvements we propose in the next section.

To address the limitations of nazief scheme, we propose the following improvements:

1) *Using a More Complete Dictionary*: we have experimented with two other dictionaries, and present our results later.
2) *Adding Rules to Deal With Plurals:* when plurals, such as "bola-bola" (balls) are encountered, we propose stemming these to "bola" (ball).

However, care must be taken with other hyphenated words such as "bolak-balik" (to and for), "berbalas-balasan" (mutual action or interaction) and "seolah-olah" (as though). For these later examples, we propose stemming the words preceding and follow the hyphen separately and then, if the words have the same root of the 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 this works for many hyphenated non-plurals.

1) *Adding Prefixes and Suffixes, and Additional Rules*:

- Adding the particle (inflection suffix) "-pun". This is used in words such as "siapapun" (where the root of the word is "siapa" (who).
- For the prefix type "ter", we have modified the conditions so that row 4 in Table 2 sets the type to "ter" instead of "none". This supports cases such as "terpercaya" (the most trusted), which has the root of word "percaya" (believe).
- For the prefix type "pe", we have modified the conditions (similar to those listed in Table 2 so that words such as "pekerja" (worker) and "peserta" (member) have prefix type "pe", instead of the erroneous "none".
- For the prefix type "mem", we have modified the conditions so that words beginning with the prefix "memp-" are of type "mem".
- For the prefix type "meng", we have modified the conditions so that the words beginning with the prefix "mengk-" are of type "meng".

2) *Adjusting Rule Precedence:*

- If a word is prefixed with "ber-" 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 where the root of the word is "masalah" (problem) and "bersekolah" (be at school) where the root of the word is "sekolah" (school).
- If a word is prefixed with "ber-" and suffixed with the derivational suffix "-an", try to remove prefix before the suffix. This solves problems with, for example, "berbadan" (having the body of) the root of the word is "badan" (body).
- If a word is prefixed with "men-" and suffixed with the derivational suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "menilai" (to mark) the root of the word is "nilai" (mark). If a word is prefixed with "di-" and suffixed with the derivational suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "dimulai" (to be started) the root of the word is "mulai" (start).
- If a word is prefixed with "pe-" and suffixed with the derivational suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "petani" (farmer) the root of the word is "tani" (farm).
- If a word is prefixed with "ter-" and suffixed with the derivational suffix "-i", try to remove prefix before the suffix. This solves problems with, for example, "terkendali" (can be controlled) the root of the word is "kendali" (control).

Fig. 1 shows a flowchart of CAT's stemming algorithm.

III. Results and Discussion

In this section, we compare the result of algorithm before and after stemming with Nazief Andriani's and CAT. CAT approach provides the easy way of stemming Indonesian language through flexibility affix classification. Therefore, the affix additional can be applied in easy way. We experiment used to test are the students' journals of Department of Information Technology of Faculty of Communication and Information Technology of Semarang University. There are 10 articles used for testing.

Based on the test results (see Table 6), it is clear that there is increasing on commonality document measurement results, the amount of which depends on how similar and not similar documents were tested with one of the documents in the database. Based on data tested above, the average of the last increase is 14% if it is done with CAT algorithm stemming.

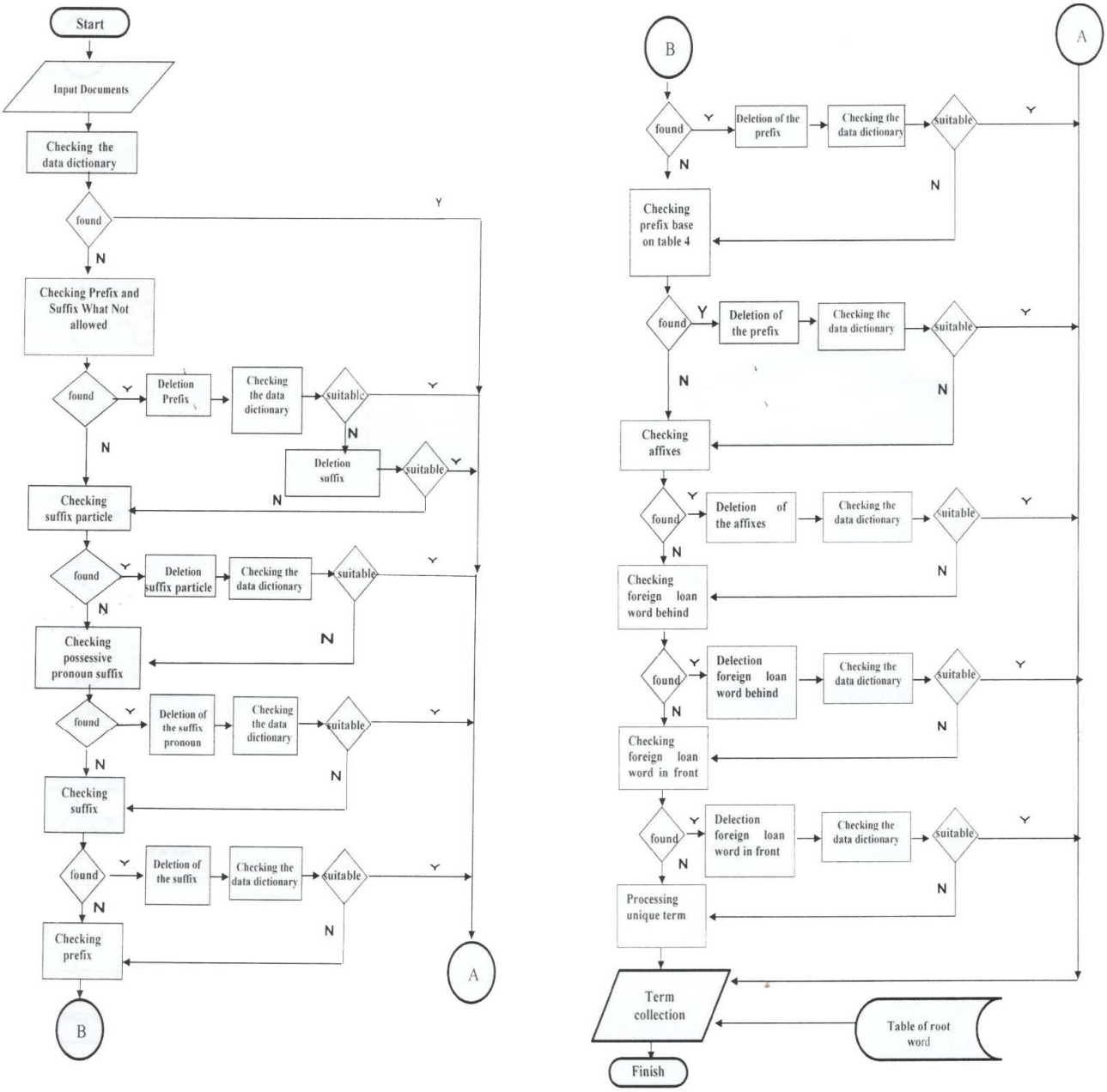

Fig. 1 Flowchart CAT's stemming algorithm

TABLE VI
THE RESULT OF STEMMING TEST FOR SIMILAR DOCUMENT WITH ID_DOC 325

Id_Document				325
Process time (second)				225.875
No	**Id_Doc**	**Without Stemming**	**Nazief Adriani's Stemming**	**CAT's Stemming**
1	264	90,000	97,000	100,000
2	298	7,406	11,751	13,508
3	292	5,927	9,981	11,257
4	313	4,472	8, 418	10,163
5	289	6,445	8,659	9,954
6	304	6,979	8, 946	9,758
7	288	7,517	8,327	9,571
8	324	6,303	8, 326	9,498
9	257	6,303	8,263	9,498
10	306	8,843	8,067	9,273

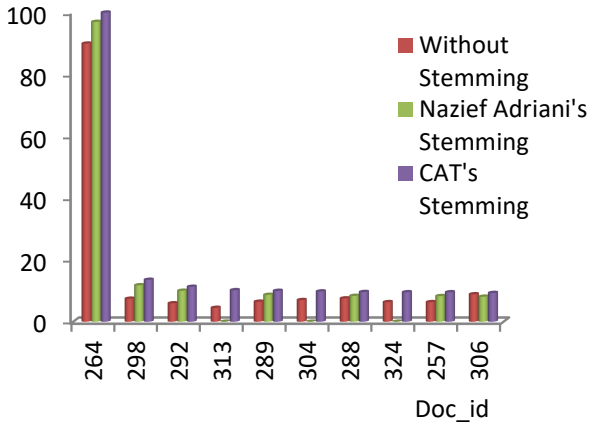

Fig. 2 Comparison results of without stemming, Nazief Adriani's stemming and CAT's stemming

V. CONCLUSION

Stemming is an important information retrieval technique. In this paper, we have investigated Indonesian stemming and presented an experimental evaluation of Indonesian stemmers. The results show that a successful stemmer is complex, and requires the careful combination of several features: support for complex morphological rules, progressive stemming of words, dictionary check after each step, trial-and-error combinations of affixes, and recoding support after prefix removal. Our results show that the new stemmer is the most effective scheme. It will increase about 14,741 % if we use the new stemmer.

We intend to continue this work. We will improve the dictionaries by curating them to remove non-root and add root words. We also plan to extend the nazief stemmer further to deal with cases where the root of the word is ambiguous.

ACKNOWLEDGMENT

We thank Bobby Nazief for providing source code and the dictionary used in this paper, Semarang University, and Gunadarma University for allowing us to use their laboratory, equipment, and instruments, as well as the experimental area.

REFERENCES

[1] Sharma, D., "Improved stemming approach used for text processing in information retrieval system", Master of Engineering in Computer Science & Engineering, Thapar University, Patiala, 2012

[2] Moral, C., Antonio, A., Imbert, R., Rmirez J., "A survey of stemming algorithms in information retrieval", Inf. Res.: Int Electron. J. **19**(1), 2014

[3] Maurya, V., Pandey, P., Maurya, L.S., "Effective information retrieval system", Int. J. Emerg. Technol. Adv. Eng. **3**(4), 787–792, 2013

[4] Singhal, A., "Modern information retrieval: a brief overview" IEEE Data Eng. Bull. **24**(4), 35–43, 2011

[5] Adriani, Mirna, Jelita Asian, Bobby Nazief, Seyed Mohammad Tahaghoghi and Hugh Williams, "Stemming Indonesian: A Confix-Stripping Approach", ACM Transactions on Asian Language Information Processing, Vol. 6, No. 4, 2007.

[6] Arifin, A. Z. & Setiono, A. N., "Classification of Event News Documents in Indonesian Language Using Single Pass Clustering Algorithm", in 'Proceedings of the Seminar on Intelligent Technology and its Applications (SITIA)', Teknik Elek- tro, Sepuluh Nopember Institute of Technology, Surabaya, Indonesia. 2002

[7] Bakar, Z. A. & Rahman, N. A, "Evaluating the effectiveness of thesaurus and stemming methods in retrieving Malay translated Al-Quran documents", in T. M. T. Sembok, H. B. Zaman, H. Chen, S.R.Urs & S. Myaeng, eds, 'Digital Li- braries: Technology and Management of Indigenous Knowledge for Global Access', Vol. 2911 of Lecture Notes in Computer Science, Springer- Verlag, pp. 653 – 662., 2003.

[8] Gustad, T. & Bouma, G, "Accurate stemming of Dutch for text classification", Language and Computers 45(1), 104–117, 2002.

[9] Oraasan, C., Pekar, V. & Hasler, L., "A comparison of summarisation methods based on term specificity estimation", in 'Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC2004)', Lisbon, Portugal, pp. 1037 – 1041, 2004

[10] Porter, M., "An algorithm for suffix stripping," Program 13(3), 130–137, 1980

[11] Savoy, J., "Stemming of French words based on grammatical categories", Journal of the Ameri- can Society for Information Science 44(1), 1–9, 1993

[12] Tala, Fadillah Z, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia", Master Thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, The Netherlands, 2003.

[13] Madenda, S, "Pengolahan Citra & Video Digital", Erlangga, Jakarta, 2015.

[14] Karczmarek, Pawe, Kiersztyn, Adam, Pedrycz, Witold and Rutka, Przemys ,"Chain Code-Based Local Descriptor for Face Recognition," Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, paper. 403 , p. 307.

[15] Y. Luo and Y. Wen and D. Tao and J. Gui and C. Xu, "Large Margin Multi-Modal Multi-Task Feature Extraction for Image Classification," IEEE Transactions on Image Processing, vol. 25, pp. 414-427, Jan. 2016.