Generating Affixed Words from a Root Word and Getting Lemma from Affixed Word in Bahasa: Indonesian Language

Andri Budiman Oktarino, Dwi Taruna Winahyu, Andrew Halim, and Derwin Suhartono

***Abstract*—Previously, there were morphological analyzer and lemmatization method for Bahasa: Indonesian language, yet they have not handled all occurred cases. Therefore, we develop an algorithm which combines two tasks; they are to generate affixed words from a root word and vice versa. The current morphological analyzer to generate affixed words has not covered in analyzing two words, whilst the current lemmatization method cannot find out the lemma from an affixed word which has confix and reduplication. Hence, we will cover these issues in order to enhance the current methods. The algorithm concerns only in Bahasa. The algorithm to generate affixed word is based on the two-level morphological analyzer, while refinement of lemmatization method is based on rule precedence and token checking. After implementing the algorithms, we find out that affixed word produced is 12.63% productive words, 86.98% non-productive words, and 0.39% incorrect words for the affixed word, whilst lemmatization can achieve 96.11% accuracy.**

***Index Terms*—Affixed word, root word, Bahasa, morphological analyzer, lemmatization.**

I. Introduction

Language is an important communication tool for human. By using language, people can communicate one to another directly. Consequently, we have to master language and its components, such as vocabulary, structure, etc. By understanding them, we can have an interaction each other without any miscommunication.

There are many researches in language and technology field. Yet, not all languages can be integrated to all kind of invented technologies since they have their own characteristics. Every language has its own rules, as well as it cannot be separated from the language convention. Therefore, if a language is going to be embedded into technology, a specific technique which focuses only to one language should be built.

Bahasa (Indonesian language) is an important language in South East Asia region. In fact, Bahasa becomes the 4th famous national language in the world [1]. Related with the issue nowadays that people use the language badly, we often find the improper usage of language in Bahasa. For example, the word "diubah" (literally means "be changed") is frequently replaced with the word "dirubah" (grammatically wrong). By understanding the language morphology, hopefully people are able to decrease the number of errors in using the language either grammatically or semantically.

In understanding the language morphology, some basic mechanism will be involved such as stemming and lemmatization. Stemming is a process that is aimed to reduce number of variation in representing a concept to be a standard or formal representation [2], whilst lemmatization is a process to seek the basic form or usually called as lemma from a particular word [3]. Stemming and lemmatization are used in defining language morphology. Morphology is knowledge about how to generate a word from the smaller units which has specific meanings. The smaller units here are called as morphemes [4].

Based on the background, an algorithm of word generator that constructs affixed word from root word and vice versa are built. The benefit of using this generator is that user can observe how an affixed word can be constructed, what are the root words, and any other information inside words. From all the information user has, hopefully it can be a new resource for user to learn about word morphology. Furthermore, it can be an initialize research to develop search engine, machine translation, or many applications related to natural language processing.

The rest of the paper is organized as follows. We first review and explore the related work in Section II. We describe the proposed method how to generate affixed words in the Section III. Experiment design, testing result and discussion will be presented in Section IV. The final section; it is Section V, will conclude and summarize the findings of our research.

II. Related Work

Stemming is used to reduce variation amount in representing a concept to be a standard or formal representation [2]. Adriani, Asian, Nazief, Tahaghoghi, Williams tried to improve the accuracy of stemming that Nazief and Adriani made before [5]. According to their analysis, errors that are existed in previous research mostly come from some aspects; they are non-root words in dictionary, incomplete dictionary, and words that are written by using hyphenation, whereas the remaining is caused by ineffective and ordering rules. By using the same dataset, the improvement has been made [6] from [5]; it reaches around 95% in accuracy. The accuracy is higher 2-3% than the previous stemming approach.

A two-level morphological analyzer was introduced by Pisceldo, Mahendra, Manurung, and Arka [7]. The

Manuscript received September 10, 2015; revised December 15, 2015. This work was fully supported by Bina Nusantara University, Jakarta, Indonesia.

A. B. Oktarino, D. T. Winahyu, and A. Halim were with Bina Nusantara University, Indonesia.

D. Suhartono is with Bina Nusantara University, Jakarta, Indonesia (e-mail: dsuhartono@binus.edu).

morphological analyzer for Bahasa is divided into two components; they are morphotactic and morphophonemic rule. The rules in each component are usually worked in parallel. Moreover, the rules are combined with vocabulary to complete the design. A word that is going to be analyzed will follow this sequence:
vocabulary → morphotactic rules → morphophonemic rules → surface

Before the result occurs in the surface, it will follow the sequential order starts from vocabulary to find out the actual morpheme of the word. After passing the vocabulary checking, the word will then be analyzed by morphotactic and morphophonemic rules. In designing a morphological analyzer, morphotactic rules are crucial to model how two or more morphemes can be merged. However, the merging process is still not completed; hence changes have to be made after these morphemes are merged. For these issues, morphophonemic rules are defined for phonetic changes that occur. After done with those processes, the analysis result will be delivered through the surface.

The research result here is a morphological analyzer for Bahasa that is able to give the detailed analysis from affixation process using two level morphological approaches. This approach is able to handle reduplication, and non-concatenative morphological process. This technique is usually called as IndMA.

There was another research which develops IndMA further, it was conducted by Larasati, Kubon and Zeman [8]. A robust finite state morphology tools for Bahasa is proposed. The technique is called as MorphInd. The research describes about morphological analyzer and lemmatization from given words so that it can be processed further. MorphInd creates morphological information which in its output format becomes morphemic segmentation, morpheme lemma position, lexical category, and morphological features. MorphInd gives better scopes rather than IndMA. MorphInd covers clitics, numeral alternation, and additional particle morphemes which were not covered by IndMA.

Related with MorphInd, a lemmatization process specifically for Bahasa is developed by Suhartono, Christiandy, and Rolando [9]. The research idea comes from the research about stemming in Bahasa [10]. The lemmatization goal is to modify the Enhanced Confix Stripping so that it can be used in accordance with the lemmatization principle. However, it has the similarity in some process inside, such as affix removal to get the lemma form. The lemmatization process includes several process; they are dictionary lookup, rule precedence check, inflectional suffix removal, derivational suffix removal, derivational prefix removal, recoding, suffix backtracking, return original word or return lemma. They are not sequential order process, but it has some conditional statement to be checked before going through the process. Based on testing result, this technique achieved probably 98% accuracy. It has to be enhanced by defining an algorithm for some words exception; such as repetitive words, words with infixes, proper nouns, abbreviations, and foreign words.

III. Proposed Method

In this research, sample data was taken from the online article such as Kompas, Detik, Tempo and printed articles from Kompas. The total tested words were 1098. The words are inputted to the algorithm scheme in figure 1 to be processed.

Fig. 1 explains the whole process of the algorithm. We represent each process by using alphabets. The description of the process is given below:

A. Database Lookup

The database contains all the root words in Bahasa and their word class. Database lookup is conducted after a word has been inputted and it occurs repeatedly in time when every affix removal happens. The first database lookup is used to determine which algorithm will be selected. If input word is not found in database, then go to lemmatization process otherwise continue to the affixed word generator process.

B. Morphotactic

Morphotactic is a process to find out which affix is suitable to be attached to input word based on its word class. Based on [7], we categorize the word class into 4 categories; they are Noun, Verb, Adjective, and Etc.

C. Morphophonemic

The morphophonemic rule contains the rules that can change the prefix pronunciation in accordance with first alphabet from the input word.

D. Return Word and Its Affixes

After done with morphotactic and morphophonemic rules, then the output are displayed. The output consists of all alternatives affixes which can be attached to the input word.

E. Token Lookup

In the lemmatization rule, if the input word is not found in the database, then the symbol "-" inside the words will be observed.

F. Repeated Word Concatenations

If the symbol "-" is existed in the word, then the words which are connected with the symbol will be concatenated. The words before and after "-" are assigned to the different variables. If both variables contain the same word, then we will do step A to find out the dictionary form of the word. Otherwise we will do affix removal rule to each variable. The variable which contains root word will not execute the rule. After the affix removal is finished to both variables, we will do database lookup again to look for their dictionary form.

G. Return Lemma Word

If the root word is still not found after execute all lemmatization rule, then input word is considered as incorrect and will be processed further, otherwise it will be displayed at the output screen.

H. Rule Precedence Checking

If the input word does not exist in the database and it does not have tokens, then do the rule precedence checking. Rule precedence is part of rules which permit affix removal; it begins with new prefix and continues to suffix [4]. As for, the requisite of rule precedence is the combination of the following affixes:

- 'be-' and '-lah'

- ‘be-’ and ‘-an’
- ‘me-’ and ‘-i’
- ‘di-’ and ‘-i’
- ‘pe-’ and ‘-i’
- ‘te-’ and ‘-i’

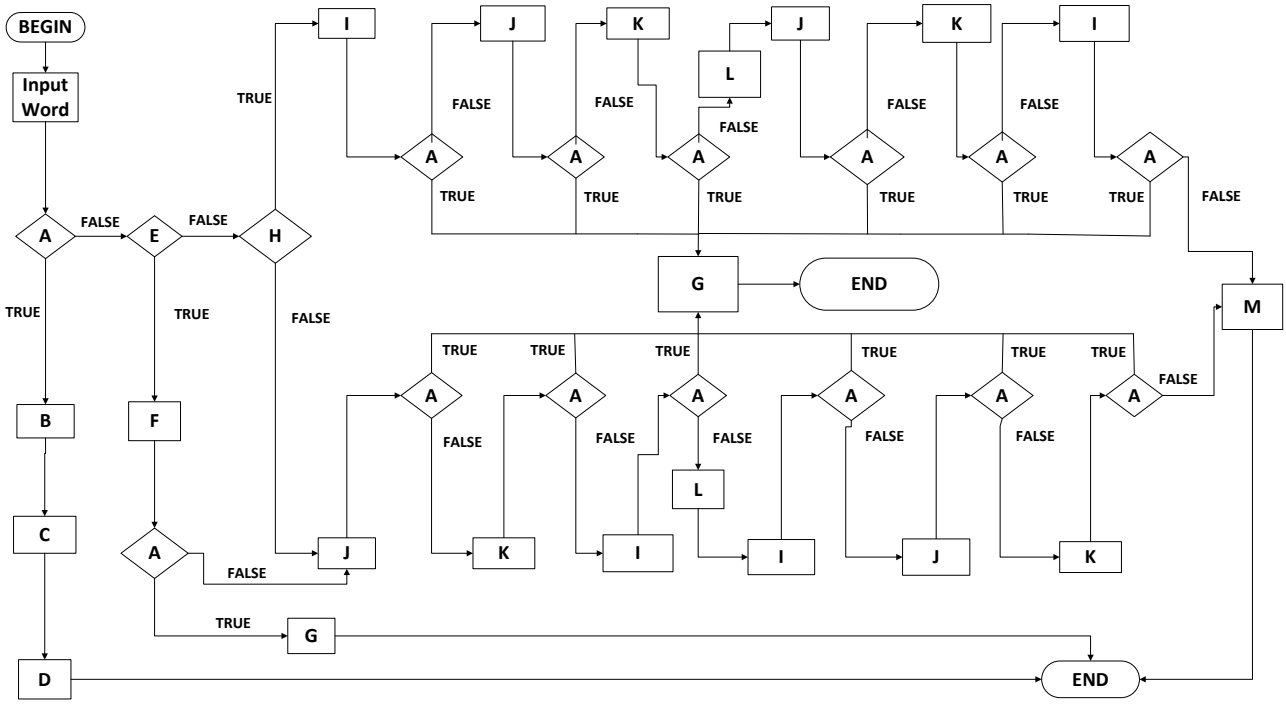

Fig. 1. Algorithm to generate affixed word and lemma word.

If the input word does not fulfill rule precedence requisite, affix removal will be executed from suffix. By applying rule precedence, there are some affix word cases which can be finished faster and achieve more accurate result. Example, for the word “bermasalah” (literally means problematic), if we remove suffix “-lah” before prefix “ber-”, then the result will not be appropriate as expected; it is “masa” (literally means period/time)

A. Derivational Prefix Removal

Derivational prefix has two groups. The first group is prefixes which have no morphophonemic, for example: ‘di-’, ke-’, ‘ku-’, and ‘se-’. Otherwise, the second group has morphophonemic form, it is the alternative form following the word’s first alphabet, for example: ‘me-’, be-’, ‘pe-’, and ‘te-’

B. Inflectional Suffix Removal

Inflectional suffix has two groups. The first group is particle suffix, for example: ‘-lah’, ‘-kah’, ‘-tah’, and ‘-pun’. The second group is suffix that explains possession, example: ‘-ku’, ‘-mu’, and ‘-nya’

C. Derivational Suffix Removal

Derivational suffix consists of ‘-i’, ‘-kan’, and ‘-an’. They are removed in this section.

D. Affixed Word Reconstruction

If lemmatization process cannot get a word that is existed in the database, then the word will be returned back as the original word like when it is being inputted. After affixed word has been reconstructed, lemmatization process will be reversely done. If in the first experiment, affix is removed from the front, then after reconstruction affix will be removed from the back, and vice versa. This is useful because there are some words which have the same form with the affixed word so that in some cases the word will be removed and database lookup will returns fail result. For example by following lemmatization rule, the word “dimakan” (literally means “be eaten”) will remove suffix, otherwise if “-kan” is removed from the word “dimakan”, database lookup will not be succeed because part of the word which should not be removed is still removed. After doing word reconstruction and the process is executed reversely (from the front), then we can get the word “makan” (literally mean “eating”).

E. Return Input Word

After all the steps in the algorithm has been conducted but the output still cannot be found, then input word will be displayed at the output screen and the input word will be considered as the wrong one.

IV. Results and Discussion

From 1098 testing words, 577 of them are root words so they are processed through affixed words generator, 437 of them are affixed words so they are processed through lemmatization, and 84 words are invalid words.

A. Getting Lemma from Affixed Word

By conducting test to the lemmatization algorithm which consists of 437 tested words, there are 17 errors occur in affix removal. It means that 420 words out of them result on success. In brief, the percentage of success and failure of the lemmatization are:

$$L(S)=\frac{420}{437}\times 100\% = 96.11\%$$

$$L(F) = \frac{17}{437} \times 100\% = 3.89\%$$

$L(S)$ = percentage of success on lemmatization

$L(F)$ = percentage of failure on lemmatization

From the calculation, percentage of success on lemmatization is very high; it is 96.11%, whilst the percentage of failure is very low; it is 3.89%.

TABLE I: COMPARISON TO OTHER APPROACHES

Research Topic	Approach	Methodology	Accuracy
Stemming Indonesian: A Confix-Stripping Approach	Based on dictionary and rules	Rule of prefix, suffix, confix by dictionary lookup	95%
Lemmatization Technique in Bahasa: Indonesian Language	Based on dictionary and rules	Rule of prefix, suffix, confix by dictionary lookup	98%
Generating Affixed Words from a Root Word and Getting Lemma from Affixed Word in Bahasa: Indonesian Language	Based on dictionary and rules	Rule of prefix, suffix, confix by dictionary lookup	96.11%

From Table I, the percentage of success in our lemmatization is lower than previous research [9] which achieves around 98% in accuracy. This is because we differs the input into 2 categories such that if the input word is the root word then it will run the affixed word generator algorithm

B. Getting Affixed Words from a Root Word

The testing uses 577 root words which results on 14647 affixed words, but not all of them are productive words. For affixed words generator, the output result is categorized into 3; they are productive words, non-productive words, and error words. From 14647 affixed words, the total of productive words is 1851, total of non-productive words is 12471, and total of error words is 55. Accordingly, the percentages for each category are:

$$M(P) = \frac{1851}{14647} \times 100\% = 12.63\%$$

$$M(NP) = \frac{12741}{14647} \times 100\% = 86.98\%$$

$$M(E) = \frac{55}{14647} \times 100\% = 0.39\%$$

$M(P)$ = percentage of productive words on affixed word generator

$M(NP)$ = percentage of non-productive words on affixed word generator

$M(E)$ = percentage of error words on affixed word generator

Productive words are words in which their way to construct has followed the rules in Bahasa and they can be found in Kamus Besar Bahasa Indonesia (dictionary of Indonesian language).

Non-productive words are words in which their way to construct has followed the rules in Bahasa but they cannot be used by common people such that it cannot be found in dictionary. Nevertheless there is probability for them to be used by people in the future.

Error words are words in which their way to construct has followed the rules in Bahasa but the resulted words are not suitable to the common words.

The percentage of productive words is quite low; it is 12.63%. Although the words are correctly constructed based on the rules but they are not commonly used by people. Therefore, the total amount between productive and non-productive words is significantly high; it is 99.61%. It means the algorithm runs well to produce affixed words.

C. Testing Summary

Based on the testing conducted, 17 errors occur in lemmatization algorithm, 7 of them are caused by over-lemmatized words, 4 of them are caused by under-lemmatized words, and 6 of them are the incorrect rule implementation. The sample of errors can be seen in Table II below.

TABLE II: EXAMPLE OF ERRORS IN LEMMATIZATION

Case	Example
Over-lemmatized	mengurangi → urang
Under-lemmatized	pengamat → kamat
Incorrect rule implementation	sesuai → sua

At the **over-lemmatized** case, the removed affixes are more than it should be, yet it still results on a word that is existed in database. For example the word "mengurangi" (literally means "decrease") should become "kurang" (literally means "deficient"). Therefore, the algorithm cut the alphabet 'k' and the word becomes "urang" (literally means "tree"). It will be claimed as the output.

For affix removal in the word "pengamat" (literally means observer), the alphabet 'ng' should be removed and it remains the word "amat". Nevertheless, the alphabet 'ng' is not removed and it is replaced by 'k' so that "amat" becomes "kamat" (literally means "wood"). It is existed in the database so "kamat" becomes the lemma of "pengamat". This is an **under-lemmatized** case.

The word "sesuai" (literally means "appropriate") should become "suai" after being lemmatized. This is not happened because the rule precedence implementation to the word "sesuai" is not suitable. According to the algorithm, the alphabet 'i' will be removed first so it remains "sesua". Furthermore, the algorithm will remove "se-" so that it remains the word "sua" (literally means "meet"). Although this is an **incorrect rule implementation**, but the word is existed in the database, so it is claimed as the output.

V. CONCLUSION

From the testing conducted to the algorithm, we conclude that:

1) By combining two kinds of algorithm; they are affixed word generator and lemmatization, we can still achieve a good accuracy as the result.
2) This research has solved the problem in the previous research. The lemmatization can handle reduplication and affix removal in affixed words in reduplication form
3) The morphotactic and morphophonemic rules are quite hard to be implemented in Bahasa. It is because many heterogeneities and inconsistencies in its language rule. Inaccuracy in this research happens because of this issue.

In fact, some of words constructions in Bahasa are based on user agreement.

ACKNOWLEDGMENTS

We would like to thank Mrs. Hari Sulastri from Agency of Language Development of Ministry of Education and Culture in Indonesia for supporting us about deeper knowledge in Bahasa. We thanks Bina Nusantara University for the sponsorship and financial support during conducting this research from the initial until the publication.

REFERENCES

[1] J. N. Sneddon, *The Indonesian Language: Its History and Role in Modern Society*, Australia: University of New South Wales Press, 2003.
[2] G. Kowalski, *Information Retrieval Architecture and Algorithms*, New York: Springer, 2011.
[3] A. K. Ingason, S. Helgadóttir, H. Loftsson, and E. Rögnvaldsson, "A mixed method lemmatization algorithm using a hierarchy of linguistic identities (HOLI)," *Advances in Natural Language Processing, Lecture Notes in Computer Science*, vol. 5221, pp. 205-216, 2008.
[4] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, New Jersey: Prentice-Hall, Inc., 2008.
[5] B. Nazief and M. Adriani. "Confix stripping: approach to stemming algorithm for Bahasa Indonesia," *Technical Report*, Faculty of Computer Science, University of Indonesia, Depok, 1996.
[6] J. Asian, H. E. Williams, and S. M. M. Tahaghoghi, "Stemming Indonesian," in *Proc. Conferences in Research and Practice in Information Technology*, vol. 38, 2005.
[7] F. Pisceldo, R. Mahendra, R. Manurung, and I. W. Arka. "A two-level morphological analyzer for the Indonesian language," *Australasian Language Technology Association Workshop*, vol. 6, pp. 142-150, 2008.
[8] S. P. Larasati, V. Kubo˘n, and D. Zeman, "Indonesian morphology tool (MorphInd): Towards an Indonesian corpus," *Systems and Frameworks for Computational Morphology*, pp. 119-129, 2011.
[9] D. Suhartono, D. Christiandy, and Rolando, "Lemmatization technique in Bahasa: Indonesian language," *Journal of Software*, vol. 9, no. 5, pp. 1-8, 2014.
[10] M. Adriani, J. Asian, B. Nazief, S. M. M. Tahaghoghi, and H. E. Williams, "Stemming Indonesian: A confix-stripping approach," *ACM Transactions on Asian Language Information Processing*, vol. 6, no. 4, pp. 1-33, 2007.

Andri Budiman Oktarino was born on October, 10, 1992. He has completed his bachelor degree from Bina Nusantara University, Jakarta, Indonesia majoring Computer Science. His interest is in the area of intelligent system.

Dwi Taruna Winahyu was born on February 24, 1992. He has completed his bachelor degree from Bina Nusantara University, Jakarta, Indonesia majoring Computer Science. His interest is in the area of intelligent system.

Andrew Halim was born on August 26, 1992. He has completed his bachelor degree from Bina Nusantara University, Jakarta, Indonesia majoring Computer Science. His interest is in the area of intelligent system.

Derwin Suhartono was born on January 24, 1988. He has completed his bachelor and master degree majoring computer science in Bina Nusantara University, Jakarta, Indonesia since 2011. Currently, he is taking his PhD study in University of Indonesia. He is a lecturer of intelligent system field in Bina Nusantara University, Jakarta, Indonesia. His previous work was as java developer. He was developing web site for banking and telecommunication in probably 1 year. His interest is in the area of natural language processing and intelligent system.