https://doi.org/10.31436/iiumej.v23i1.1783

IN-IDRIS: MODIFICATION OF IDRIS STEMMING ALGORITHM FOR INDONESIAN TEXT

FEBIARTY WULAN SUCI, NUR HAYATIN* AND YUDA MUNARKO

Department of Informatics, Engineering Faculty,
University of Muhammadiyah Malang, Malang, Indonesia

**Corresponding author: noorhayatin@umm.ac.id*

(Received: 11th January 2021; Accepted: 18th March 2021; Published on-line: 4th January 2022)

ABSTRACT: Stemming has an important role in text processing. Stemming of each language is different and strongly affected by the type of text language. Besides that, each language has different rules in the use of words with an *affix*. A large number of the words used in the Indonesian language are formed by combining root words with *affixes* and other combining forms. One of the problems in Indonesian stemming is having different types of *affixes*, and also having some *prefixes* that changes according to the first letters of the root words. Implementing Idris stemmer for Indonesian text is of interest because Indonesia and Malaysia have the same language root. However, the results do not always produce the actual word, because the Idris algorithm first removes the prefix according to Rule 2. This elimination directly affected the Idris stemmer result when implemented to Indonesian text. In this study, we focus on a modified Idris stemmer (from Malay) to IN-Indris with Indonesia context. In order to test the proposed modification to the original algorithm, Indonesian online novels excerpts are used to measure the performance of IN-Idris. A test was conducted to compare the proposed algorithm with other *stemmers*. From the experiment result, IN-Idris had an accuracy of approximately 82.81%. There was an increased accuracy up to 5.25% when compared to Idris accuracy. Moreover, the proposed stemmer is also running faster than Idris with a gap of speed of around 0.25 seconds.

ABSTRAK: Stemming mempunyai peranan penting dalam pemprosesan teks. Stem setiap bahasa adalah berbeza dan sangat dipengaruhi oleh jenis bahasa teks. Selain itu, setiap bahasa mempunyai peraturan yang berbeza dalam penggunaan kata dengan awalan. Sebilangan besar kata-kata yang digunakan dalam bahasa Indonesia dibentuk dengan menggabungkan kata akar dengan afiks dan bentuk gabungan lain. Salah satu masalah dalam bahasa Indonesia adalah mempunyai pelbagai jenis awalan, dan juga mempunyai beberapa awalan yang berubah sesuai dengan huruf pertama kata dasar. Menerapkan stemder Idris untuk teks Indonesia adalah minat kerana Indonesia dan Malaysia mempunyai akar bahasa yang sama. Namun, hasilnya tidak selalu menghasilkan kata yang sebenarnya, kerana algoritma Idris pertama kali menghapus awalan menurut Peraturan 2. Penghapusan ini secara langsung mempengaruhi hasil batang Idris ketika diterapkan ke teks Indonesia. Dalam kajian ini, kami memfokuskan pada stemmer Idris yang diubahsuai (dari bahasa Melayu) ke IN-Indris dengan konteks Indonesia. Untuk menguji cadangan pengubahsuaian pada algoritma asli, petikan novel dalam talian Indonesia digunakan untuk mengukur prestasi IN-Idris. Ujian dilakukan untuk membandingkan algoritma yang dicadangkan dengan stemmer lain. Dari hasil eksperimen, IN-Idris mempunyai ketepatan sekitar 82,81%, ada peningkatan ketepatan hingga 5,25% dibandingkan dengan ketepatan Idris. Selain itu, stemmer yang dicadangkan juga berjalan lebih cepat daripada Idris dengan jurang kelajuan sekitar 0.25 saat.

KEYWORDS: *Idris stemming; IN-Idris; NLP; text pre-processing*

1. INTRODUCTION

Finding the root words of derivative words is still a challenge, especially for text processing. Stemming is a process to transform derivative words into root words. Stemming methods are widely applied in Natural Language Processing (NLP) and also text mining especially when involved in extracting information that is used to find new information from various written sources [1], such as sentiment analysis [2]. Stemmer, an algorithm for stemming, has an important role in text processing because the results of stemming are used to extract features in the text. For the text mining area, stemmer is important in the initial stage, which has a function to convert unstructured text into structured representative formats that can then be processed by machines [3]. Implementing stemmer for text processing has proven to improve the performance of text pre-processing [4]. It also has an impact on instance classification results using Indonesian Quran translation [5], and reduces the different forms of a word into a root word [6].

Stemming of each language is different, and it is strongly influenced by the type of text language. Moreover, each language has different rules in the use of words with *affixes* [7]. However, for some languages stemming can be achieved by applying the appropriate morphological rules [8]. Indonesian used Bahasa Indonesia as a formal language. In the Indonesian text, words are formed from morphological rules [9]. A large number of the words used in the Indonesian language are formed by combining root words with *affixes* and other combining forms [10]. Indonesian has stemming problems that are specific to the language. One of the problems is having different types of *affix*, another is having some *prefixes* that change according to the first letters of the root words. For example, the *prefix* "*me-*" becomes "*mem-*" when attached to a root word starting with the letter "*b-*" as in "*mem-buat*" (to make, in English), but it becomes "*meny-*" when attached to a root word starting with the letter "s-" as in "*meny-*[s]*impan*" (to store, in English) [11].

In Indonesian text processing, some researchers used a stemming algorithm such as Nazief and Adriani (N&A) [12], Confix Stripping Algorithm (CS) [13], and Enhanced Confix Stripping (ECS) [14], which are developed for Indonesian text. However, several researchers also implemented *stemmers* from other languages such as Idris stemmer (for Malay language) [15], and Potter stemmer (for English) [16]. In this study, we focus on Idris stemmer. This is interesting because Indonesian and Malay have the same language root. Malay has similarities with the Indonesian language, so the Idris algorithm can also be used to process Indonesian texts [17].

Earlier research compared the processing time and accuracy of the Nazief and Adriani (N&A) stemmer with the Idris stemmer [8]. The research applied to Indonesian text to know which algorithm was better. For the evaluation, the researcher used five story texts to compare the stemming result. The result presented that the N&A stemmer obtained a 97% accuracy with a processing time of approximately 0.03 seconds per word, while the Idris stemmer reached an accuracy of 91% but needed around 0.02 seconds per word for processing. Another research focused on analysing stemmer strength in Indonesian text documents based on the parameters of the *icf* and *wc* values, and also analysed the level of accuracy and speed based on word results. As a result, Idris stemmer was accurate and succeeded in producing root words from Indonesian text, but it still had shortcomings namely, increasing the possibility of *overstemming*. The results of stemming are not appropriate because the Idris algorithm first removes the *prefix* according to Rule 2. Elimination directly refers to the Rule 2 effect on the results of stemming, where the

algorithm does not always result in the actual word. Moreover, other experiments [8] have proven that the Idris algorithm has an advantage in the speed of the stemming process but the level of accuracy is still low.

To obtain a better level of accuracy and faster process for the stemming, this study proposed to modify the Idris algorithm with Indonesian text. We measured the accuracy and the speed of the stemming process as a result of the modification of the Idris algorithm. The modified algorithm is called IN-IDRIS.

2. RELATED WORK

2.1 Indonesian Stemmers Algorithm

The Nazief and Adriani (N&A) stemmer [12] is based on morphological rules that are interlinked and grouped, then encapsulate the allowed part of the word and do not include *affixes* such as *prefixes*, *suffixes*, and *confixes* to get the root of a word. The performance of this algorithm is based on three parts, they are grouping *affixes*, usage rules and the establishment of limits, and use of the dictionary. The dictionary becomes an important part because it is used to check whether a word has met its stem or not. Before the *affix* removal process, several things must be considered in the uses of this algorithm, such as *inflexion suffixes*, derivation *suffixes*, derivation *prefixes*, and *prefix* disallowed *suffixes* [18]. The performance of the N&A approach is the most complex approach.

Confix Stripping (CS) algorithm is an Indonesian stemmer [6] that was developed based on improvement from the Nazief & Adriani algorithm. CS stemmer added an additional stemming step called *LoopPengembalianAkhiran* (final return loop). This algorithm was done by adding some *prefix* rules and modifying *prefix* rules, particularly adding rule precedence. From comparing performance results, it is shown that the CS algorithm has better performance than the N&A stemmer [13]. The development of the CS stemmer is the Enhanced Confix stripping (ECS) [14]. From the experiment, the result showed that the ECS stemmer succeeded in increasing the accuracy from the modification of the ECS stemmer using a non-deterministic method that was able to identify the possibilities of root words that can be formed in a single word through the candidate list [19].

An Indonesian stemmer based on the dictionary is the VEGA stemmer. This algorithm used rule sets to determine whether an *affix* can be removed from a word. The rules are accessed in the order they are presented in the code. When one rule fails, the algorithm proceeds to the next. A major shortcoming of the VEGA approach is the absence of a lookup stage where words are only compared to a dictionary of known root words. Stemming continues as long as the word contains *affix* letters, often leading to *overstemming*. Moreover, the algorithm does not cater to cases where recoding is required. Finally, the reliance on strict rules necessitates that the rules be correct and complete, and prevents ad hoc restoration of *affix* combinations [20].

2.2 Idris Stemmer Algorithm

The Idris stemmer is developed by Idris et al. that is designed for the Malay language. The Malay language has similarities with the Indonesian language [16], so this algorithm is also applicable to Indonesian texts. This algorithm has two dictionaries (general and local dictionaries) in determining stemming results [17]; where the local dictionary contained a list of root words in the Malay history vocabularies. This algorithm implements only two patterns of the rules which are *prefix* and *suffix* rules. By using only two patterns of *affixes* that are *prefix* and *suffix*, it can reduce the numbers of the rule sets.

https://doi.org/10.31436/iiumej.v23i1.1783

Idris algorithm first checks the words against *prefix* rules then checks the words against the *suffix* rules. If not, many errors such as *overstemming* can occur [15]. Idris algorithm adopts Arifin and Setiono's assumption that each Indonesian word has two *prefixes* and three *suffixes* [21].

Idris algorithm applied a different recoding scheme and progressive stemming. The algorithm checks the dictionary after each step, stopping and returning the stemmed word if it is found. The scheme works as follows: first, after checking for the word in the dictionary, the algorithm tests if the *prefix* of the word matches a *prefix* that may require recoding; second, if recoding is required, it is performed and the resultant word is searched for in the dictionary, while if recoding is not required, the *prefix* is removed as usual and the word is checked in the dictionary; third, having removed a *prefix*, *suffix* removal is attempted and, if it succeeds, the word is checked in the dictionary; and, last, the algorithm returns to the second step with the partially stemmed word. There are two variants of this algorithm: the first changes *prefix* and then performs recoding, while the second does the reverse [22]. The Idris algorithm description is referred to in Table 1.

Table 1: The description of Idris algorithm

ALGORITHM 1: Idris Algorithm
1. Check the word in the dictionary. If the word is found, then the word is considered as the root word and exit. Otherwise, proceed to the next step.
2. Check the word in the *prefix* rules. If the word matches the *prefix* rules, check the *prefix* pattern and the first letter of the stem word. Otherwise, go to step 7.
3. If the *prefix* pattern matches the pattern in Rule 2, then apply Rule 2 to the word. Otherwise, remove the *prefix* and go to step 6.
4. Check the *prefix* of the word, adjust the pattern in Rule 2. If it matches the fourth rule, then check the stem word in the dictionary and proceed to the next step. Otherwise, remove the *prefix* and go to step 6.
5. If the word is not in the dictionary, then return to step 4, otherwise, remove the *prefix* and proceed to the next step.
6. Check the word in the dictionary. If the word found, then the word is considered as the root word and exit. Otherwise, proceed to the next step.
7. Check the words in the *suffix* rules. If it matches with the *suffix* rules, then remove

Prefixes usually give rise to spelling variations and exceptions in the root word. Errors may occur during the *stemming* process where the stemmed word is not complete. The Idris algorithm has another rule called Rule 2 where it can be applied to the *prefix* removal only [15]. The rule is if the word *stemming* is not complete, checking the first letter of the word and if the word starts with a vowel, then:

1. Add *t* after removing *men-* or *pen-*
2. Add *k* after removing *meng-* or *peng-*
3. Add *s* after removing the *meny-* or *peny-*
4. Add *f* or *p* after removing *mem-* or *pem-*

3. MODIFICATION OF IDRIS ALGORITHM

In this section, we explain the proposed modification of the Idris algorithm, called IN-Idris. From the results of experiments and analysis of the Idris stemmer, there are some

https://doi.org/10.31436/iiumej.v23i1.1783

examples of words that are not appropriate. For example, the word "*memasukkan*" (to enter, in English) found the root word "*pasuk*". The word is in the dictionary and the algorithm will consider it as the root word, but the word "*pasuk*" is not meant, and the actual root word is "*masuk*" (enter, in English).

The results of stemming are not appropriate because the Idris algorithm first removes the *prefix* according to Rule 2. In the example, after removing the *prefix* "*mem-*" the word "*asuk*" is obtained, so the fourth rule in Rule 2 after removing the prefix "*mem-*" then add the letter *p*, then it becomes the word "*pasuk*". Elimination directly refers to the Rule 2 effect on the results of stemming, where the algorithm does not always result in the actual word.

The process occurs because at the 2nd stage, the algorithm only checks whether or not the words are in accordance with the *prefix* rules and Rule 2 patterns without *prefix* removal and if at the 3rd stage if the checked words are in accordance with Rule 2, the algorithm immediately applied the rule to the word. After analysing the results of stemming, the next step is to make improvements by changing the process from stages 2 and 3. The proposed improvements are as follows:

1. In step 2 the algorithm will first check the word in the *prefix* rule, if appropriate then the *prefix* will immediately be removed.
2. In step 3 if checking the *prefix* rule is not appropriate, the algorithm will check the *prefix* pattern in Rule 2 and the first letter of the input word. If it is in accordance with Rule 2, the algorithm will apply the rule to the word.

The modification of the Idris algorithm is described in Table 2 below:

Table 2: The description of IN-Idris algorithm

ALGORITHM 2: IN-Idris Algorithm
1. Check the word in the dictionary. If the word is in the dictionary, then the word is considered as the root word and exit. Otherwise, proceed to the next step.
2. Check the word in the *prefix* rule. If the word matches the prefix rules, then remove the *prefix* and go to step 6. Otherwise, proceed to the next step.
3. Check the *prefix* pattern and the first letter of the word to be stemmed. If the *prefix* pattern matches Rule 2, then apply Rule 2 to the word. Otherwise, remove the *prefix* and go to step 6.
4. Check the *prefix* of the word, adjust the pattern in Rule 2. If it matches the fourth rule, then check the stem word in the dictionary and proceed to the next step. Otherwise, remove the *prefix* and go to step 6.
5. If the word is not found in the dictionary, then return to step 4, otherwise, remove the *prefix* and proceed to the next step.
6. Check the word in the dictionary. If the word found, then the word is considered as the root word and exit. Otherwise, proceed to the next step.
7. Check the words in the *suffix* rules. If it matches with the *suffix* rules, then remove the *suffix* and go to step 1. Otherwise, just go to step 1.

From Fig.1 we can see the step-by-step of the proposed algorithm depicted through the flowchart. The processes with grey colour use the modified rule from Idris stemmer.

https://doi.org/10.31436/iiumej.v23i1.1783

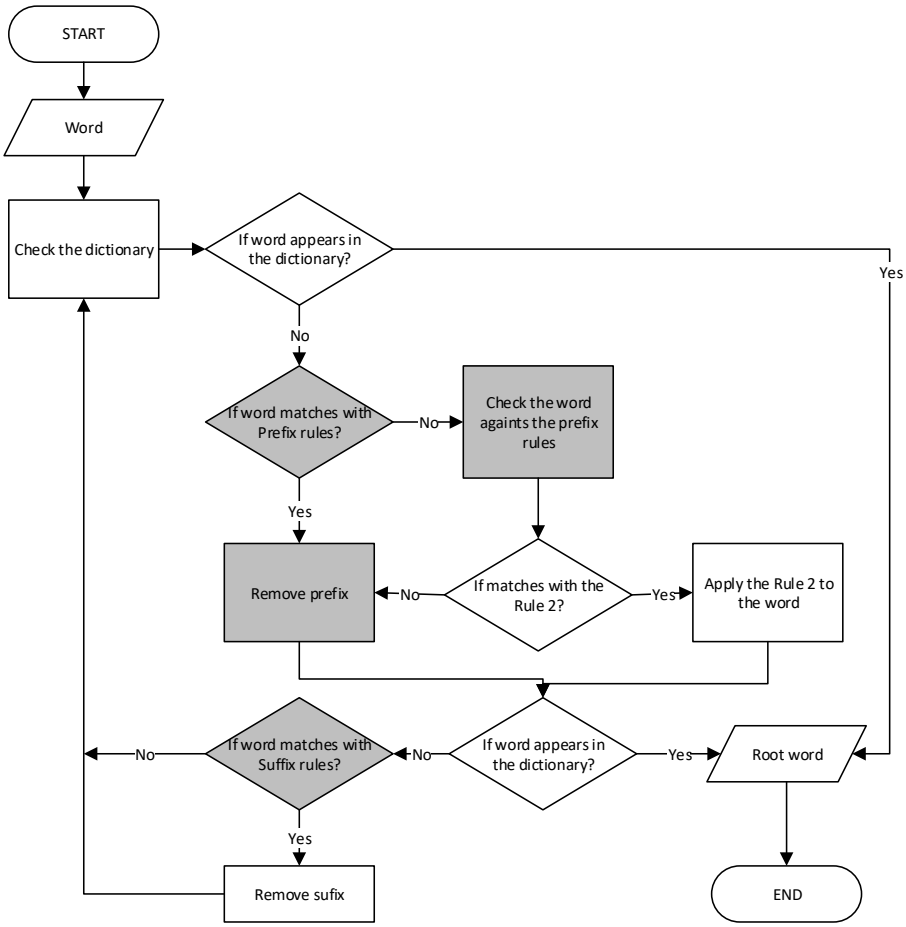

Fig. 1: The flowchart of modified Idris algorithm (IN-Idris).

4. EXPERIMENT AND EVALUATION

4.1 Dataset

For the experiment process, this study used a dataset retrieved from Indonesian online novel excerpts to measure the performance of the proposed algorithm. The novel data were sourced from research by Permatasi [6] which have a total of 6 novels, and there were 4 novels from our collection so that there is a total of 10 novels to be used. Novel excerpt data represented as a document so that a total of the document is 10 documents for the evaluation process. Novel excerpt data used was the only content without using the title of the novel. The dataset needs to go through the *preprocessing* stage so that it can be processed to the next stage. Meanwhile, the dictionary of root words is used based on Kamus Besar Bahasa Indonesia (The Big Indonesian Dictionary, in English). Kamus Besar Bahasa Indonesia (KBBI) is the most complete Indonesian dictionary which is popularly used as a reference in Indonesian text processing.

4.2 Evaluation Scenarios

The research evaluation was done by comparing the *stemming* results of the modification of the Idris algorithm (IN-Idris) and other *stemming* algorithms (original Idris, N&A, ECS). In this experiment, we analysed the accuracy and the speed of every stemmer. Testing scenarios that were carried out tested 10 Indonesian novels excerpts. Generated output was in the form of stemming root words, followed by counting the number of true and false stemming results from each algorithm. Several steps were used,

including case folding, tokenization, eliminating *stopwords*, and *stemming*. Case folding is the process of changing all letters to lowercase and removing punctuation, numbers, and certain symbols [9]. After that, the tokenization process is breaking a stream of text up into phrases, word, symbols, or other elements [23]. Then, the words taken from tokenizing results will be processed to eliminate the unimportant words that are included in the *stopwords* list. Furthermore, each document content will be stemmed. The *stemming* process is then carried out to process the *affix* words into root words.

The results will be assessed to produce the number of both true words and wrong words; so that the accuracy value of both *stemmers* can be calculated. This evaluation also measured the speed of processing time during changing the root word from the stem word of the document. Eq. (1) [8] is used to calculate the accuracy value of *stemming* result from each document d_i where i =1 to 10. Furthermore, t represented the total of true words from d_i while w is the total of the wrong words from d_i and the word total is represented by n.

$$Accuracy = \left(\frac{t-w}{n}\right) x100\% \quad (1)$$

4.3 Experiment Results and Discussion

The first experiment compared the *stemming* result of the Idris and the IN-Idris stemmer. The testing results are shown in Table 3, which describes the experimental results for every document tested. The table also presents the number of words contained in each document, the number of true and wrong words, and also the speed and accuracy value of both *stemmers*. For both Idris and IN-Idris *stemmers*, the maximum accuracy resulted from document 2, which changed from 83.13% to 87.95%. On the other hand, the minimum accuracy was produced in document 7 *stemming* results with around 69.49% for the IN-Idris and 67.01% for the Idris *stemmer*.

Table 3: Comparison accuracy and speed of both Idris and IN-Idris Stemmer

Doc	Number of words	Idris				IN-Idris			
		True Word	Wrong Word	Speed (s)	Accuracy (%)	True Word	Wrong Word	Speed (s)	Accuracy (%)
1	239	97	27	3.40	78.23	106	17	3.32	86.18
2	261	120	28	3.62	81.08	127	21	3.02	85.81
3	323	138	28	3.79	**83.13**	146	20	3.80	**87.95**
4	511	188	46	**5.28**	80.34	192	42	**5.96**	82.05
5	608	233	79	7.96	74.68	257	55	8.81	82.37
6	2224	912	230	**42.16**	79.86	967	173	**47.17**	84.82
7	1361	522	257	25.56	**67.01**	539	237	18.52	**69.46**
8	1210	415	137	14.71	75.18	443	109	13.59	80.25
9	794	361	91	12.95	79.87	384	65	12.90	85.52
10	997	366	114	18.61	76.25	400	78	18.40	83.68
Average				**13.80**	**77.56**			**13.55**	**82.81**

On the other hand, in terms of speed, *stemming* based on IN-Idris needed around 13.55 seconds. However, this is faster than processing root words with the Idris *stemmer* which has an average time of around 13.80 seconds. In general, based on the researcher's experiment (shown in Table 3), it could be concluded that documents that have under 600 words just need under 9 seconds for *stemming*. Meanwhile, if the number of the words is

https://doi.org/10.31436/iiumej.v23i1.1783

under 1300, it will take under 25 seconds to process. Furthermore, for the documents that have over 2000 words, over 47 seconds are required.

Figures 2 and 3 show the accuracy and speed comparison from both *stemmers* represented in a graph. Based on Fig. 2, in general, it showed that the accuracy value produced by IN-Idris was higher than the Idris *stemmer* result. The average value of accuracy from the IN-Idris *stemmer* was approximately 82.81% while *stemming* using the Idris algorithm showed an accuracy of just 77.56%. IN-Idris performance dominated for whole documents tested and reached an accuracy of over eight in ten.

Generally, IN-Idris presented a faster speed than Idris in stem word processing (shown in Figure 3). However, Idris showed good speed when processing documents 4, 5, and 6. In terms of accuracy, IN-Idris was better than Idris when processing those documents. The average speed of the proposed algorithm reached approximately 13.55 seconds while Idris needed 13.80 seconds on average for *stemming*. As a result, the speed gap between the IN-Idris and the Idris *stemmers* is around 0.25 seconds.

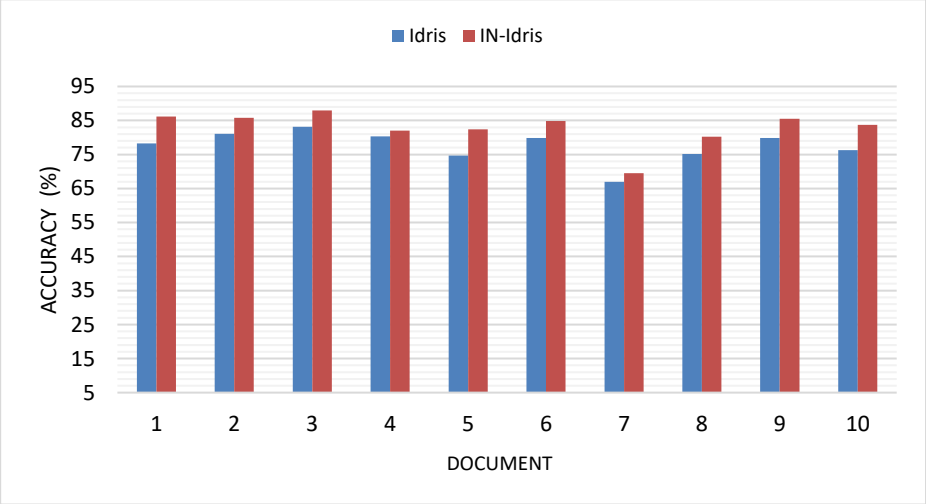

Fig. 2: The graph visualization of accuracy comparison between Idris and IN-Idris *stemming*.

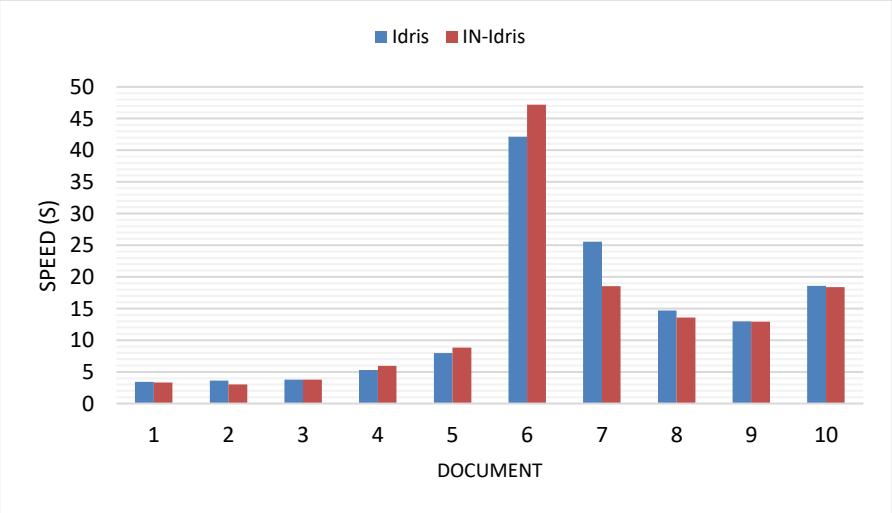

Fig. 3: The graph visualization of speed between Idris and IN-Idris *stemming*.

https://doi.org/10.31436/iiumej.v23i1.1783

Table 4 shows the sample text of the *stemming* result from the Idris and the IN-Idris *stemmers*. The table contained: stem words (the original word taken from the novel collections), the root word for ground truth (the word which is used as a reference for evaluation), and the *stemming* result from both stemmers. The *stemming* result columns present which stem words were successfully changed to their root word and also show which were unsuccessful. There are some cases where IN-Idris was more powerful than Idris, as proven from cases no. 1, 2, 4, 5, 6, 7, 9, and 10. However, for case no. 3 both algorithms could not work properly. Some cases have shown that the Idris *stemmer* affected character's reduction, such as root word "pendek" changed to be "dek", and the root word "tetes" changed to be "tes".

Table 4: Sample text from *Stemming* result

Case no.	Stem Word	Root Word Ground truth	In English	Stemming Result	
				IDRIS	IN-IDRIS
1	*memusnahkan*	*musnah*	destroyed	*memusnahkan*	*musnah*
2	*percayai*	*percaya*	believe	*percayai*	*percaya*
3	*kelahiranmu*	*lahir*	born	*kelahiranmu*	*kelahiranmu*
4	*menyeramkan*	*seram*	scary	*menyeramkan*	*seram*
5	*menyakitkan*	*sakit*	sick	*menyakitkan*	*sakit*
6	*mengerikan*	*ngeri*	horrified	*gerik*	*ngeri*
7	*pendekku*	*pendek*	short	*dek*	*pendek*
8	*terulur*	*ulur*	stretch out	*ulut*	*ulut*
9	*tetesan*	*tetes*	drops	*tes*	*tetes*
10	*menuruni*	*turun*	down	*urun*	*turun*

In a further experiment, we compared the result of the IN-Idris *stemmer* with two other algorithms (N&A and ECS) in addition to the previous Idris result. Table 3 presented the accuracy and speed of those four *stemmer* results when tested using 10 documents selected. From the table, the accuracy of ECS *stemmer* in both upper and lower accuracy is around 90.37% and 80.36% respectively. On the other hand, the result of N&A *stemmer* has a maximum accuracy of 89.86%, and the minimum accuracy is just 73.78%. Meanwhile, the accuracy of both Idris and IN-Idris have been described and presented in the previous table. Overall, the accuracy results showed significant performance over eight in ten.

In terms of speed, Table 5 described that the fastest time needed by the ECS *stemmer* is approximately 2.41 seconds while the longest time is around 30.50 second. This result is not much different from the N&A result that has a speed in maximum and minimum around 36.68 seconds and 2.51 seconds, respectively. Meanwhile, Idris has around 42.16 seconds as its lowest speed and around 3.40 seconds as its fastest speed. On the other hand, IN-Idris has the fastest and longest speeds, approximately 3.02 seconds and 47.17 seconds, respectively.

In Fig. 4, the graph presented a comparison of the accuracy of the four *stemming* algorithms. Overall, the accuracy results showed significant performance over eight in ten. Whereas Idris produced lower accuracy, which just reached 77.56% inaccuracy, and the ECS *stemmer* has the highest accuracy of 85.92%. Meanwhile, the proposed *stemmer,* IN-Idris, signified accuracy in 82.81%, which is lower than both the ECS and N&A *stemmers*, but the accuracy gap was small with just around 3%.

Table 5: Comparison accuracy and speed result between Idris and IN-Idris *Stemmer*

Doc	N&A		ECS		Idris		IN-Idris	
	Speed (s)	Accuracy (%)	Speed (s)	Accuracy (%)	Speed (s)	Accuracy (%)	Speed (s)	Accuracy (%)
1	2.51	89.43	2.73	90.37	3.40	78.23	3.32	86.18
2	3.19	89.86	3.14	87.73	3.62	81.08	**3.02**	85.81
3	4.37	88.55	3.55	85.75	3.79	83.13	3.80	87.95
4	5.87	86.32	2.41	82.58	5.28	80.34	5.96	82.05
5	7.35	84.94	6.71	87.66	7.96	74.68	8.81	82.37
6	36.68	86.93	30.50	88.46	42.16	79.86	47.17	84.82
7	21.66	73.78	13.73	87.00	25.56	67.01	18.52	69.46
8	19.45	86.59	9.59	85.49	14.71	75.18	13.59	80.25
9	12.65	86.49	12.98	85.61	12.95	79.87	12.90	85.52
10	11.97	87.45	16.62	80.36	18.61	76.25	18.40	83.68
Average	**12.57**	**85.61**	**10.20**	**85.92**	**13.80**	**77.56**	**13.55**	**82.81**

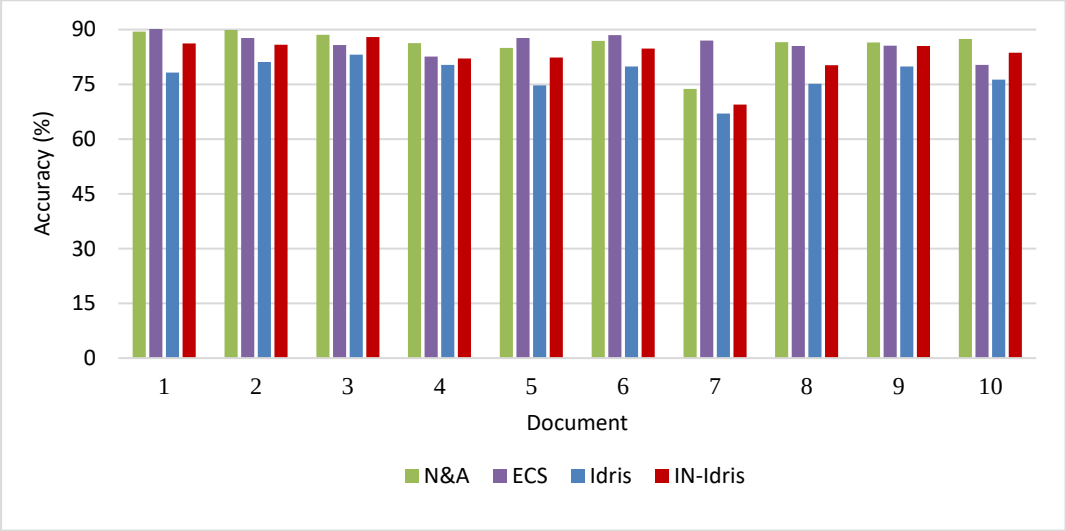

Fig. 4: The graph visualization of accurate comparison results for both stemmers.

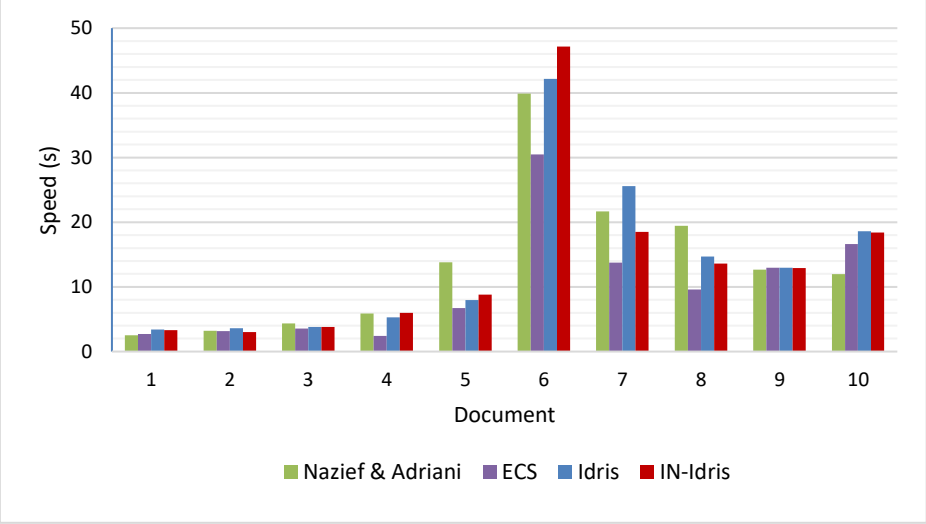

Fig. 5. The graph visualization of accurate comparison results for both stemmers.

Figure 5 depicts a comparison of the *stemming* speed of the four algorithms. *Stemming* processed by ECS was the fastest, with just 10.20 seconds, and the longest *stemming* needed approximately 13.80 seconds when it was processed through the Idris *stemmer*. On the other hand, the proposed method IN-Idris has a duration time of around 13.55 seconds to transform words into their root. This result is proof that IN-Idris needs extra time than N&A and ECS in *stemming*. However, there is one instance when IN-Idris finished *stemming* faster than the others, it is shown for document 2, where just needed 3.02 seconds.

Based on the experiment, the accuracy value is affected by the true word total produced for each document of *stemming* results. The higher the number of true words, the higher accuracy produced. However, the smaller the number of true words, the smaller the resulting accuracy. Furthermore, the result of the *stemming* accuracy for each document is different for each *stemmer*. It can be caused by the different rules implemented in each algorithm, which can be called a *stemmer* characteristic.

On the other hand, in terms of speed, the ECS *stemmer* is faster than the Idris algorithm. That result is a refutation for the previous research [6] which was conducted to test the accuracy of the Idris and ECS *stemmers* implemented in Indonesian text documents. That research has claimed that Idris has an excellent performance in speed. We concluded that the speed processing in *stemming* related to the number of words contained in each document, where the time needed to do the *stemming* process is relatively based on the tool or programming language used [24].

It was found that the Idris algorithm had a low accuracy value caused by an *affix* cutting error in the *stemming* process that affected the results. The error was related to the application of the *prefix* removal first rule on the Idris *stemming*. This caused root words that had the same letters as a *prefix* to be discarded, and it caused un-matching and errors in the final result. This error could be corrected by IN-Idris that obtained good *stemming* results with an increased accuracy until 5.25% from the Idris *stemmer*. However, in the modification of the Idris algorithm, IN-Idris still had a deficiency that caused words to fail in *stemming*. There is a shortcoming that IN-Idris could not handle, such as the *affix* "*se-ku*" on the word "*sebelahku*" (next to me, in English) whereas with the Idris algorithm, the word "*sebelahku*" was changed to "*belah*".

The challenge to develop a good *stemming* algorithm for future work is how a *stemmer* can be able to handle some problems including deletion error particularly occurring due to typing errors and mistakes in place names, people's names, and also foreign languages. The most powerful *stemmer* is the ECS algorithm, and some of its rules can be adopted to improve IN-Idris performance. However, the ECS *stemmer* applied the *suffix* deletion rule first and checked the dictionary every time it removed the *affix*. This can lead to word *overstemming* because the *affix* removal process was carried out as much as possible according to the applied rules, thus affecting the accuracy results. Besides that, there is need for further development that can modify a combination of *prefix* and *suffix* rules to produce a better-*stemming* performance.

5. CONCLUSION AND FUTURE WORK

IN-Idris is a new *stemmer* for Indonesian text that was inspired from and improved upon the Idris Malay *stemmer*. Based on the experiment, the researchers found that the Idris algorithm has a low accuracy value caused by affix cutting errors on the *stemming* process which affected the results. This error could be corrected by IN-Idris so that it can

obtain good *stemming* results with an increased accuracy until 5.25% from Idris *stemmer*. IN-Idris performance dominated for all documents tested, and overall the accuracy results showed significant performance of over eight in ten. In terms of speed, IN-Idris presented a faster speed than Idris in stem word processing. Finally, we conclude that the accuracy value is affected by the total of true words produced for each *stemming* result document. The higher the true words, the higher accuracy produced. However, the smaller the true words, the smaller the accuracy that resulted.

REFERENCES

[1] Vijayarani DS, Ilamathi, MJ., Nithya M. (2015) Preprocessing techniques for text mining - An overview. J. Computer Science & Communication Networks, 5(1): 7-16.
[2] Buntoro G, Arifin R, Syaifuddiin G, Selamat A, Krejcar O, Hamido F. (2021) The Implementation of the machine learning algorithm for the sentiment analysis of Indonesia's 2019 Presidential election. IIUM Engineering Journal, 22(1): 78-92.
[3] Nassirtoussia AK, Aghabozorgia S, Wah TY, David CLN. (2014) Text mining for market prediction: A systematic review. Expert Systems with Applications, 7653-7670.
[4] Rizki AS, Tjahyanto A, Trialih R. (2019) Comparison of stemming algorithms on Indonesian text processing. Telkomnika, 17(1): 95-102.
[5] Utomo FS, Suryana N, Sanusi Azmi, M. (2020) Stemming Impact analysis on Indonesia Quran translation and their tafsir classification for ontology instances. IIUM Engineering Journal, 21(1): 33-50.
[6] Permatasari N. (2016) Analisis Perbandingan algoritma Idris dan algoritma enhanced confix stripping (ECS) stemmer pada dokumen teks bahasa Indonesia. Universitas Komputer Indonesia. https://repository.unikom.ac.id/130/
[7] Titin W, Kerami J, Arief S. (2017) Determining Term on text document clustering using algorithm of enhanced confix stripping stemming. International Journal of Computer Applications, 157(9): 8-13.
[8] Prasidhatama A, Suryaningrum KM. (2018) Perbandingan algoritma Nazief & Adriani dengan algoritma Idris Untuk pencarian kata dasar. Jurnal Teknologi & Manajemen Informatika, 4(1): 1-4.
[9] Prihatini PM, Putra ID, Giriantari IAD, & Sudarma M. (2017) Stemming Algorithm for Indonesian Digital News Text Processing. International Journal of Engineering and Emerging Technology, 2(2): 1-7.
[10] Mena VV, Saputri K. (2018) Contrastive analysis between English and Indonesian prefixes and suffixes in the descriptive texts of student's textbooks. English Community Journal, 2(1): 175-182.
[11] Jelita A. (2007) Effective Techniques for Indonesian Text Retrieval. Melbourne: RMIT University.
[12] Adriani M, Asian J, Nazief B, Tahaghoghi SMM, & Williiams HE. (2007) Stemming Indonesian: A confix-stripping approach. ACM Transactions on Asian Language Information Processing (TALIP), 6(4): 1-33.
[13] Widayanto H & Huda AF. (2017) Comparison Nazief Adriani and CS stemmer algorithm for stemm real data. In e-Proceeding of Engineering, 4(3): 5215. Bandung, Indonesia.
[14] Arifin AZ, Mahendra IPAK, & Ciptaningtyas HT. (2009) Enhanced Confix stripping stemmer and ants algorithm for classifying news document in Indonesian language. in International Conference on Information & Communication Technology and Systems. In The International Conference on Information & Communication Technology and Systems, 5:149-158. Surabaya, Indonesia.
[15] Idris N, Mustapha SS. (2001) Stemming For Term Conflation In Malay Texts.
[16] Porter, M. F. (1980). An algorithm for suffix stripping. Program1, 14(3): 130-137.
[17] Paramitha ES. (2012) Analisis dan implementasi stemming menggunakan algoritma Idris pada dokumen teks berbahasa indonesia. Telkom University, Indonesia.

[18] Mardiana T, Adji TB, Hidayah I. (2016) Stemming Influence on Similarity Detection of Abstract Written in Indonesia. Telkomnika, 14(1): 219-227.
[19] Rifai W, Winarko E. (2019) Modification of Stemming algorithm using a non deterministic approach to Indonesian text. Indonesian Journal of Computing and Cybernetics Systems, 13(4): 379-388.
[20] Vega VB. (2001) Information retrieval for the Indonesian language. Master's thesis, National University of Singapore.
[21] Arifin AZ & Setiono AN. (2002) Klasifikasi dokumen berita kejadian berbahasa Indonesia dengan algoritma single pass clustering. In Prosiding Seminar on Intelligent Technology and its Applications (SITIA). Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.
[22] Asian J, Williams HE, & Tahaghoghi SMM. (2005) Stemming Indonesian. In Proceedings of the Twenty-eighth Australasian conference on Computer Science, 38: 307-314. Australia.
[23] Winarti T, Kerami J, Arief S. (2014) Tokenization and Filtering process in rapidminer. International Journal of Applied Information Systems, 7(2): 16-18.
[24] Zaman B. (2014) Modifikasi algoritma Porter untuk stemming pada kata bahasa Indonesia. In Seminar Nasional Teknologi Informasi dan Komunikasi (SENTIKA 2014), 543-550. Surabaya, Indonesia.