JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 5 No. 1 (2024) 91-103
https://publisher.uthm.edu.my/ojs/index.php/jscdm

KAROMA: Karonese Morphological Analyzer Based on Graph Theory

Ichwanul Muslim Karo Karo[1,2], Mohd Farhan Md. Fudzee[2]*, Shahreen Kasim[2], Azizul Azhar Ramli[2]

[1] *Computer Science, Faculty of Mathematics and Natural Science, Medan State University, Medan, 20221, INDONESIA*
[2] *Faculty of Computer Sciences and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, MALAYSIA*

*Corresponding Author: farhan@uthm.edu.my
DOI: https://doi.org/10.30880/jscdm.2024.05.01.008

Article Info

Received: 1 December 2023
Accepted: 25 April 2024
Available online: 21 June 2024

Keywords

KAROMA, graph theory, member checking, natural language processing (NLP)

Abstract

A morphological analyzer is essential for increasing the quality of natural language processing (NLP) research in national and local languages. Karonese is a local language of Karo ethnics from north Sumatra, Indonesia. Karonese terms have unique phonology, exhibiting variations in spellings and pronunciations while retaining the same meaning and time. Several NLP studies with Karonese case studies have limited access to Karonese morphology analyzers. This study aims to suggest a Karonese morphological analyzer based on graph theory (KAROMA). The KAROMA idea adopts a word-based morphological approach whereby the Karonese terms are expressed in a completed graph. The outcome's set of completed graphs then comprises the Karonese WordNet and is compiled for use as KAROMA. This study also provides two KAROMA evaluators: member checking-based and text similarity-based by modified cosine similarity. The KAROMA evaluation process involves synthetic sentences of Karonese to calculate its text similarity. As a result, KAROMA can detect the uniqueness of Karonese terms and normalize them. The performance of KAROMA is 99% based on member-checking and 97.16% of text similarity-based. Therefore, KAROMA has emerged as a research finding that could be applied as a Karonese stemming and lemmatization technique for a variety of NLP challenges. Furthermore, the evaluator serves as a research contribution that other researchers can apply.

1. Introduction

In the current decade, Natural Language Processing (NLP) has been one of the fastest-growing areas of research. Most NLP applications, such as text-to-speech or speech-to-text, sentiment analyzers, machine translation, spell checkers, and search engines, are not only designed to accommodate international or national languages. It has expanded to local languages, such as machine translation for Javanese, Sundanese, Bambara, Ewe, etc., and sentiment analyzers for local languages. It is certainly a challenge to develop NLP applications for a particular language to recognize the structure and grammar of that language.

A morphology analyzer is a method for identifying, structuring, and investigating the total set of possible relationships contained in words [1]. Morphological analyzers have proven to be vital subsystems of several NLP applications [1], [2], [3], [4]. A morphology analyzer is designed to work exclusively with a single language. In

other words, a morphological analyzer is not designed to be a grammar solver for multiple languages, and a morphological analyzer dedicated to a particular language is a requirement. Recently, morphological analyzers for local languages have been proposed to support NLP applications, such as Tamil [5], Telugu [6], Malayalam[7], etc. It shows that the morphology analyzer has also been widely developed to support NLP applications for local languages.

Karonese is a local language of Karo ethics from north Sumatra, Indonesia. Karonese has a unique phonology. A Karonese term can exhibit variations in spellings and pronunciations while retaining the same meaning [8]. Table 1 presents an example of the uniqueness of Karonese. The term "corn" could be pronounced and written in several letters (*jaung ; \jong ; \joung*) in Karonese. There are similar but not identical cases in other languages. English defines the word "know" in three forms of pronunciation based on the present, past, and future (know, knew, known), Arabic has fourteen verb pronunciations (*fi'il mudhari*) based on the subject. Morphology analyzer could be used as a stemmer or lemmatization to solve multiple spelling and pronunciation words in English, Arabic [9], Uzbek [4], Malayalam [7], etc. Previous NLP research for Karonese has been limited by text preprocessing techniques, such as the unavailability of the Karonese Stopword library, morphology analyzer [10], [11] and the inability of machine translation to interpret the cultural terms of Karonse [12], [13], [14].

Table 1 *Example of uniqueness Karonese terms*

English word	Karonese word
Corn	*Jaung ; Jong*
Don't	*Ola; Oula; Ula;*
There is/there are	*Lit ; let*

Based on the description of facts and problems related to NLP research for Karonese, this research aims to propose a morphology analyzer capable of identifying the uniqueness of Karonese terms and their functions as stemming and lemmatization. The morphology analyzer is called the Karonese Morphological Analyzer Based on Graph Theory (KAROMA). There are several possible approaches used for KAROMA, among them word-based morphology. In addition, this research also presents morphology analyzer evaluators: member checking-based and text similarity-based by modified cosine similarity. The availability of KAROMA is a research finding that can contribute to text preprocessing techniques in NLP research for Karoneses, such as sentiment analysis, text summarization, text modeling, etc. In addition, both morphology analyzer evaluators can be adopted to evaluate morphology analyzers for other languages.

2. Related Studies

Morphological analysis is a field of linguistics that studies the structure of words. It identifies how a word is produced through the use of morphemes. A morpheme is a basic unit of the English language. The morpheme is a word's smallest element with grammatical function and meaning. Free morpheme and bound morpheme are the two types of morphemes. A single free morpheme can become a complete word.

There are two types of morphology: Inflectional Morphology and Derivational Morphology [1]. Both of these types have their significance in various areas related to the NLP. Inflectional morphology is the study of processes, including affixation and vowel change, that distinguish word forms in certain grammatical categories. Inflectional morphology consists of at least five categories, provided in the following excerpt from Language Typology and Syntactic Description: Grammatical Categories and the Lexicon. As the text explains, derivational morphology cannot be easily categorized because derivation isn't as predictable as inflection. Examples are cats, men etc. Derivational Morphology is defined as morphology that creates new lexemes, either by changing the syntactic category (part of speech) of a base or by adding substantial, nongrammatical meaning or both. On the one hand, derivation may be distinguished from inflectional morphology, which typically does not change category but rather modifies lexemes to fit into various syntactic contexts; inflection typically expresses distinctions like number, case, tense, aspect, and person, among others. On the other hand, derivation may be distinguished from compounding, which also creates new lexemes, but by combining two or more bases rather than by affixation, reduplication, subtraction, or internal modification of various sorts. Although the distinctions are generally useful, in practice applying them is not always easy.

This research has identified that there are four methods for performing morphological analysis [15]: morpheme-based morphology (or an item and arrangement approach), lexeme-based morphology (or an item and process approach), word-based morphology (or a word and paradigm approach), and machine learning approach.

J. of Soft Computing and Data Mining Vol. 5 No. 1 (2024) p. 91-103

2.1 Morpheme-based Morphology

Morpheme-based morphology is a concept in which it is assumed that word formation rules may operate over morphemes [4]. It is assumed that new words are formed by applying a word formation rule to an already-existing word. On the other hand, word forms are analyzed as arrangements of morphemes. There are four main kinds of word formation: prefixes, suffixes, conversions, and compounds. A morpheme is defined as the minimal meaningful unit of a language. In a word such as independently, the morphemes are said to be in-, de-, pend, -ent, and -ly; pend is the (bound) root, and the other morphemes are, in this case, derivational affixes. In words such as "dogs," the root is "dog," and the -s is an inflectional morpheme.

Many researchers have proposed a morphology analyzer based on morpheme-based. A paper by [16] proposed a morphology analyzer for Urdu text. They defined a system for stemming the word, prefix stripping, infix stripping, postfix stripping, and word normalization. They used a machine learning algorithm to evaluate the proposed morphology analyzer for Urdu sentiment analysis. The results show that the complete Urdu stemming process gives maximum results compared to the previous process. Precision, recall, and F-1 are 89%, 93.8%, and 91.3%, respectively.

Another study proposed lemmatization for Icelandic by using suffix substitution rules derived from a large morphological database to lemmatize tagged text [17]. They called it Nefnir, the new open-source lemmatizer for Iceland. The evaluation shows that for text that is tagged correctly, Nefnir obtains an accuracy of 99.55%, and for text that is tagged with a POS tagger, the accuracy obtained is 96.88%.

Morpheme-based morphology is effective as a morphology analyzer not only for national languages but also for local languages. A study by [18] modified the Nazief and Adriani algorithm for stemming Javanese words. The first step is to collect data and create a basic word dictionary. They modified Java by removing the affix. Based on the investigation, there are three kinds of Javanese affixes: prefix, insert, and suffix. To evaluate the modified stemmer, 366 words were tested. The results show that the accuracy of modified stemmers is 95.9%. Other work by [2] provides the Kokborok morphological analyzer, and the accuracy is 72%. Table 2 presents a brief research summary of a morphology analyzer using morpheme-based.

Table 2 *Brief of morphology analyzer using morpheme-based*

Related work	Language	Proposed	Result
[4], [19]	Uzbek	Stemming	There is a system
[16]	Urdu	Stemming	precision, recall and F-1 are 89 %. 93.8 % and 91.3%
[17]	Icelandic	Lemmatization	Accuracy 99.55%
[18]	Javanese	Stemming	Accuracy 95.9%.
[2]	Kokborok	Stemming	Accuracy 72%
[20]	Tigrinya	Stemming	Accuracy 83.9%

2.2 Lexeme-based Morphology

Lexical morphology is the branch of morphology that concerns the lexicon or is rule-based [15]. Lexemes are noun, verb, and adjective stems [19]. These items manifest without exception as sound-meaning pairings that refer to something in the real world. Several studies have proposed a morphology analyzer for a language using the lexeme-based approach. A study proposed lexicon-free stemming for Kazakh by stemming the base of the word to find specific words in documents [21]. The proposed Tamil morphological analyzer was used to conduct research [22]. The rule-based approach has been successfully used in developing many natural language processing systems with an accuracy of 83%. Another study by [23] found that their system has high precision (95–98%). The disadvantage of using rule-based approaches is that if one rule fails, it will affect the entire rule that follows; that is, each rule works on the output of the previous rule.

2.3 Word-based Morphology

A word-based approach relies on the concept of words and their connection [24]. This connection can be represented not only by semantic correlation but also by phonological similarity. Therefore, a word-based approach is based on the connection of surface words to their bases. This connection is divided into two levels: phonological and semantic. A study by [25] mentions that word-based models are suitable for "representing words with identical parts but different overall meanings". A word-based approach is considered to be the main pillar of the Word and Paradigm Model (WP).

Many previous studies used this approach to solve unique morphology cases and even stated that they had better results compared to other approaches [5]. A study by [25] presents a morph analyzer for Gujarati. They have discussed a paradigm-based approach for morphological analysis of various POS tags. The algorithmic development yields a noun analysis accuracy of 84.50%, a verb analysis accuracy of 81.50%, and an adjective

analysis accuracy of 80.50%. This study agrees with the argument that the word-based approach is possibly better utilized. Word-based morphology also creates semantic algorithms for multilingual languages (Uyghur, Kazakh, and Kirghiz) [1]. The study claims that the proposed stemmer outperforms another previous stemmer.

2.4 Other Approach

Another approach to the proposed morphology analyzer is adopting a machine learning algorithm [5]. This strategy aims to find every potential candidate by using grammatical rules and an annotated corpus. ML algorithm is employed to determine the most probable morphological deconstruction for a given word. This idea has been implemented by [6], [26]. They used the SVM algorithm to provide Tamil and Telugu morphological analyzers. The accuracy of their system is 95%, even when using the identical algorithm. The Malayalam morphological analyzer by [7] is very effective, and after learning, it predicts correct grammatical features even for words that are not in the training set. Table 3 contains a research summary of the morphological analyzer based on a machine learning (ML) algorithm.

Table 3 *Brief of a morphological analyzer using ML algorithm*

Related Work	Language	ML algorithm	Result
[26]	Tamil	SVM	Accuracy and F1 score are 98 %
[6]	Telugu	SVM	Accuracy 97%
[7]	Malayalam	SVM	Accuracy 80%
[27]	Amharic	Decision Tree + Bagging	Accuracy 69.39 %
[28]	Arabic	Hidden Markov Model	Accuracy 91%
[9]	Arabic	Hidden Markov Model	Accuracy 95%

3. Proposed Method

There are two main tasks in this research. These are KAROMA and the evaluator. KAROMA was inspired by a graph and a completed graph so that it could be compiled into an algorithm. Meanwhile, the evaluator aims to evaluate the performance of KAROMA in identifying unique Karonese terms and functioning as a stemmer and lemmatizer. The evaluators adopt a member-checking approach based on dictionary and text similarity.

3.1 Graph Theory

***Lemma* 1:** Let G be a graph $G = \{V, E\}$, where V is the vertices or nodes, and E is the edges connecting the vertices/nodes.

Based on the definition of ***lemma 1***, let vertices $V = \{A, B, C, D\}$ and Fig. 1 is an example of graph $G = \{V, E\}$ that can be formed. There are four vertices and four edges that connect the vertices. These are edge AC, BC, CD, and BD. A graph edge can express a relationship. Relationships are autonomous but consistent. Imagine that vertices A, B, C, and D are cities, and edges represent road connectivity between these cities. Edges AC identifies that city A has road connectivity to city C but not to city B.

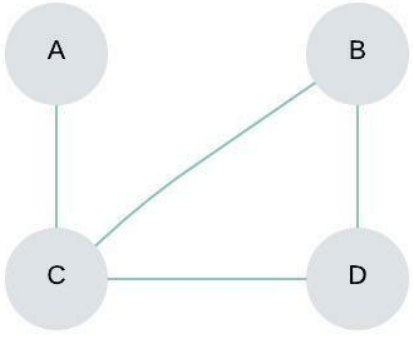

Fig. 1 *Example of graph*

J. of Soft Computing and Data Mining Vol. 5 No. 1 (2024) p. 91-103

***Lemma* 2:** A complete graph is a graph in which an edge connects each pair of graph vertices. The complete graph with n graph vertices is denoted by K_n and has $\binom{n}{2} = \frac{n(n-1)}{2}$ (the triangular numbers) undirected edges, where $\binom{n}{k}$ is a binomial coefficient.

Suppose four vertices {A, B, C, D} exist. By referring to lemma 2, a completed graph of these vertices is shown in Fig. 2. In real-world applications, the vertices can be social media users, and the edges are friendships among themselves. In other words, the four people are mutual friends on social media.

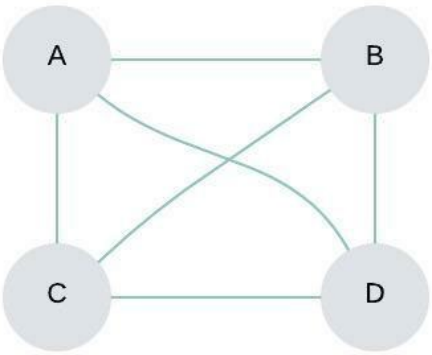

Fig. 2 *Example of a completed graph*

3.2 Proposed KAROMA Algorithm

Refer to lemma 1 and lemma 2, this study defines a completed graph with another rule. Suppose vertices (V) are Karonese terms and the edge (E) represents a condition in which Karonese terms exhibit variations in spellings and pronunciations while retaining the same meaning and time. Mathematically expressed by lemma 3**.**

Lemma 3: $K = \{V, E\}$, where V are Karonese terms and E is the condition which states that the Karonese term exhibits variations in spellings and pronunciations while retaining the same meaning and in time.

Suppose the Karonese terms are {*'ise', 'isei', 'isai', 'mejile', 'jile', 'mejilei', 'mejilai'*}, these are vertices. Based on the Karonese grammar, the terms {*'ise', 'isei', 'isai'*} have conditions that state that the Karonese terms exhibit variations in spellings and pronunciations while retaining the same meaning and in time. Hence, a completed graph will be formed with the vertices. Similarly, for the Karonese terms {*'mejile', 'jile', 'mejilei', 'mejilai'*}. Therefore, two completed graphs are constructed (shown in Fig. 3).

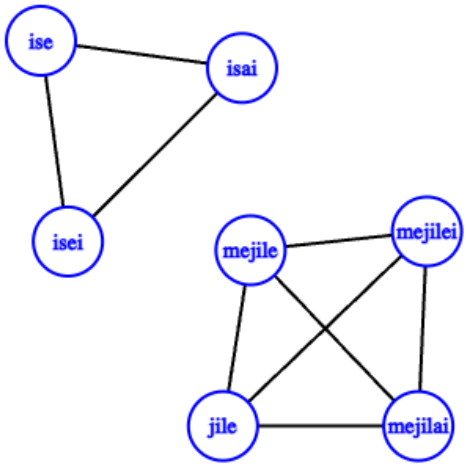

Fig. 3 *Example of a completed graph representing a Karonese term that exhibits variations in spellings and pronunciations while retaining the same meaning*

Furthermore, this study succeeded in tokenizing and identifying numerous Karonese terms from social media posts, such as statuses, tweets, comments, replies, etc. Many Karonese terms fulfill the conditions that state that the Karonese terms exhibit variations in spellings and pronunciations while retaining the same meaning and in time. The implication is that many completed graphs are formed from the vertices and edges. Thus, the set of completed graphs is created and called Karo WordNet. This Karo WordNet is used as part of the KAROMA algorithm.

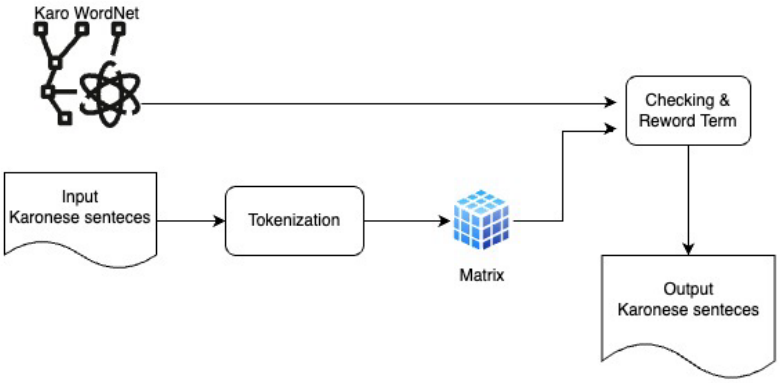

Fig. 4 *Workflow KAROMA*

The workflow of the KAROMA algorithm is presented in Fig. 4. The process starts by inputting the Karo sentence. At the same time, the system will load Karo WordNet. The algorithm will tokenize Karonese sentences. The output of the process is in Karonese terms. The terms are stored in a matrix. The next step is to check its existence in Karo WordNet. If the terms are available in Karo WordNet and in a completed graph, then the terms that exhibit variations in spelling and pronunciation while retaining the same meaning will be reworded to have the same spelling. Thus, KAROMA will output Karonese sentences that do not contain variations in spelling. Further, the proposed KAROMA algorithm can be seen below.

Proposed KAROMA Algorithm
Input: minimum two Karonese sentences

```
Load Karo WordNet
Tokenize (Karonese sentences)
matrix n(term) x 1
for i = 1 to n(term) do
 for j = 1 to number member set K
  if term(i) in K(j) then
     matrix (i) = j;
  end if.
 end for.
end for.
for i = 1 to n(term) do
 for j = 1 to n(term) do
  if term (i) and term(j) in Karo WordNet K(matrix (i)) then
   term (i) <- term (j);
  end if
 end for
end for
print (Karonese sentences)
Output :  Karonese sentences
```

To illustrate how the Karonese morphology analyzer works, Table 4 demonstrates process by process from input to output. For example, there are two Karonese sentences, '*aula kam merawa*' and '*ula kam merawa*', which are used as input. The next process is tokenizing both sentences. Thus, each term is checked on Karo WordNet. If there is the term of the sentences in Karo WordNet and the other terms (in case: same subset Karo WordNet) are on other sentences, swap it. KAROMA outputs the sentences that have been successfully identified.

J. of Soft Computing and Data Mining Vol. 5 No. 1 (2024) p. 91-103

Table 4 *Illustration of the KAROMA process*

Input	Tokenization process	Karo WordNet		Output
aula kam merawa	*'aula', 'kam', 'merawa'*	***'aula'**, 'kam', 'merawa'*	***'aula'**, 'kam', 'merawa'*	*aula kam merawa*
ula kam merawa	*'ula', 'kam', 'merawa'*	***'ula'**, 'kam', 'merawa'*	***'aula'**, 'kam', 'merawa'*	*aula kam merawa*

3.3 Member Checking Evaluator

This work proposes member checking on the dictionary as a KAROMA evaluator. It operates by taking two Karonese terms (a pair) from the corpus as input. It could be on the same K or not. The transcriber recognizes both terms (actual observe) as well as KAROMA. Suppose the results of KAROMA identification are in line with the actual observation. In that case, KAROMA is successful in identifying Karonese terms that can exhibit variations in spellings and pronunciations while retaining the same and in time. If the two results are different, KAROMA has failed, as the pseudocode of the proposed member-checking algorithm shows.

Proposed Member Checking Algorithm

```
Input:
Karonese term_1 and Karonese term_2
Actual observe (Karonese term_1, Karonese term_2)
Load Karo Wordnet
Statuses <- false;
KAROMA <- false;
for i = 1 to m do
 if term_1 and term_2 in K(i)
   statuses <- true;
end for;
if statuses == Actual observe (term_1, term_2)
  KAROMA <- true;
  return KAROMA
Output : True or False
```

3.4 Text Similarity-based Evaluator

Generate previous research studies inspired the concept of evaluating KAROMA based on text similarity [29]. They evaluate the Arabic stemmer using five distance measurements (Euclidean Distance, Cosine Similarity, Jaccard Coefficient, Pearson Correlation Coefficient, and Averaged Kullback-Leibler Divergence). Hamming Distance is another distance function that has been used to evaluate Arabic stemmers [30]. Other morphology analyzers have used the same approach but with a different distance function. They used the Minimum Distance method to evaluate Bangla morphological stemming [3]. Cosine similarity is a more representative and widely used similarity method for text analysis among the many distance measurement methods [31].

Table 5 *Example of Karonese synthetic sentences*

K	*SS*
1	***lit** denga nakan i rumah*
	***let** denga nakan i rumah*
2	*aku la pet man **rimo** macik*
	*aku la pet man **rimou** macik*
	*aku la pet man **rimau** macik*

Firstly, this study provides Karonese synthetic sentences (*SS*) (shown in Table 5). *SS* is object testing of the KAROMA algorithm in solving the problem of Karonese terms, which can exhibit variations in spellings and pronunciations while retaining the same meaning in time. For K^1, two Karonese terms can exhibit variations in spellings and pronunciations while retaining the same meaning, and in time (n = 2), these are *'lit'* and '*let*', thus

denoted by K_2^1. The implication is that there are two Karonese synthetic sentences in K_2^1, these are SS_1^1 = '*lit dengan nakan i rumah*' and SS_2^1='*let denga nakan i rumah*'. Likewise, it was for K_3^2.

Secondly, this study proposed modified cosine similarity (*ModCosine*) as a text similarity-based evaluator KAROMA. The idea is to modify the rule of cosine similarity (Equation (1)). The cosine value can be calculated using Equation (2). The value of *ModCosine* is one, indicating that the KAROMA could detect Karonese terms that exhibit variations in spellings and pronunciations while retaining the same and in time. Meanwhile, if it has a value of 0, KAROMA has failed.

$$ModCosine\left(SS_j^i, SS_k^i\right) = \begin{cases} 1;\ Cosine\left(SS_j^i, SS_k^i\right) = 1 \\ 0;\ Cosine\left(SS_j^i, SS_k^i\right) \neq 1 \end{cases} \tag{1}$$

$$Cosine\left(SS_j^i, SS_k^i\right) = \frac{SS_j^i \,.\, SS_k^i}{\left\|SS_j^i\right\|\left\|SS_k^i\right\|} \tag{2}$$

Thirdly, the KAROMA algorithm's performance is calculated using the text similarity matrix of *K*. The example of a text similarity matrix for K_n^1 is shown in Table 6. The accuracy of KAROMA on K_n^i was calculated by using Equation (3). Furthermore, if the number of i is *m*, the final accuracy of KAROMA (acc. KAROMA) is the average of each calculation set.

Table 6 *Text similarity matrix for* K_n^1

	SS_1^1	SS_2^1	...	SS_n^1
SS_1^1	$ModCosine\left(SS_1^1, SS_1^1\right)$	$ModCosine\left(SS_1^1, SS_2^1\right)$	...	$ModCosine\left(SS_1^1, SS_n^1\right)$
SS_2^1	$ModCosine\left(SS_2^1, SS_1^1\right)$	$ModCosine\left(SS_2^1, SS_2^1\right)$	...	$ModCosine\left(SS_2^1, SS_n^1\right)$
.	.	.	...	.
.	.	.		.
.	.	.		.
SS_n^1	$ModCosine\left(SS_n^1, SS_1^1\right)$	$ModCosine\left(SS_n^1, SS_2^1\right)$		$ModCosine\left(SS_n^1, SS_n^1\right)$

$$Acc.\ Karoma\ in\ K_n^i = \frac{\sum_{j=1}^{n}\sum_{k=1}^{n} ModCosine\left(SS_j^i, SS_k^i\right)}{n^2} \tag{3}$$

4. Result and Discuss

Karonese sentences are collected from three social media sites, including the post and comment on Facebook, the tweet and reply, and the comment on YouTube. In other words, Karonese sentences from other social media have not been accommodated. Generating completed graphs as Karo WordNet and the KAROMA algorithm are implemented with NLTK Python. Meanwhile, the KAROMA algorithm performs testing against Karonese synthetic sentences and the Karonese term sets.

Table 7 *Example of Karonese term sets*

K	Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and in time
1	*{'rimo','rimau','rimou'}*
2	*{'lit','let'}*
3	*{'aula','ula','ola','oula'}*
.	.
.	.
.	.
211	*{'ue', 'uei' , 'uai'}*

J. of Soft Computing and Data Mining Vol. 5 No. 1 (2024) p. 91-103

4.1 Karo WordNet

This study successfully identified 211 Karonese term sets that exhibit variations in spellings and pronunciations while retaining the same meaning and in time from social media. The example of Karonese term sets is shown in Table 7. The Karonese term set of K_3^1 defines three Karonese terms and as the first completed graph. Likewise for K_3^{211}, there are three Karonese terms. This means that there are 211 completed graphs $\left(K_n^1 - K_n^{211} \right)$ that were successfully generated. A snippet of the Karo WordNet visualization is shown in Fig. 4. The entire completed graph is referred to as the Karonese WordNet and becomes a library of KARAMA. Thus, if Karonese terms exhibit variations in spellings and pronunciations while retaining the same meaning, then the KARAMA can detect and reword the term.

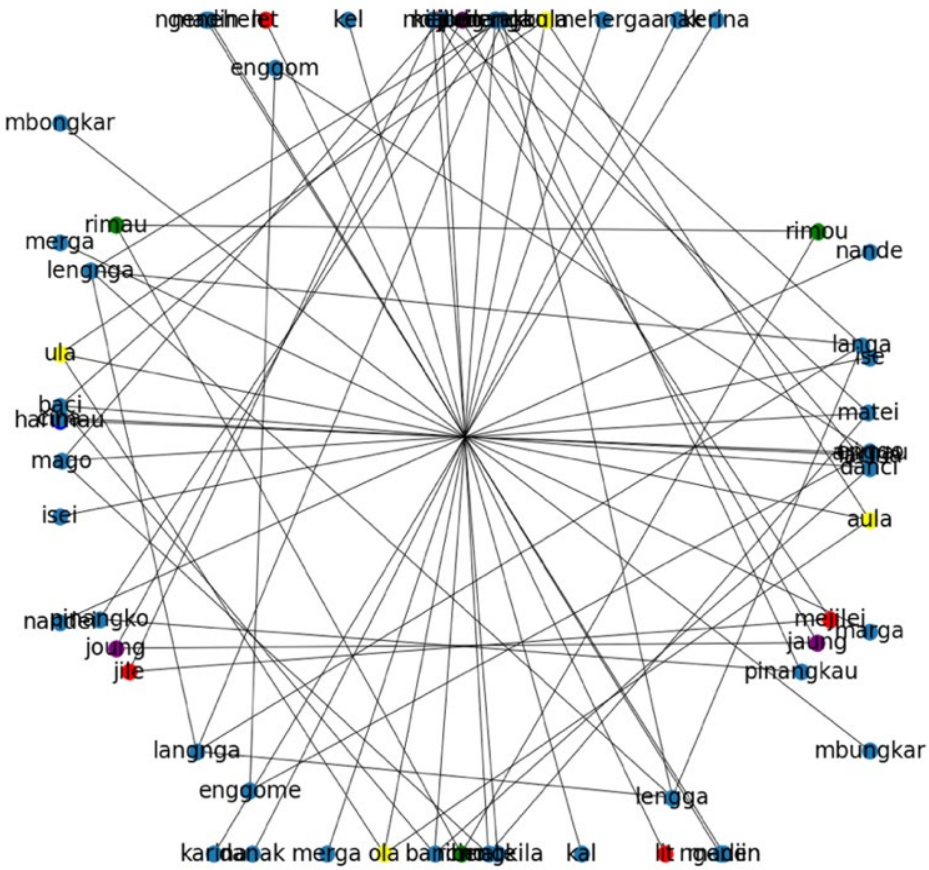

Fig. 5 *Subset Karonese WordNet*

4.2 Evaluated KAROMA using Member-Checking

This section describes and examines the performance of the KAROMA algorithm based on the member-checking approach. This study designed 100 Karonese term pairs as input terms to test the KAROMA algorithm. The transcribers have previously verified the term pairs. Not all of the input is a term that varies in spelling and pronunciation while maintaining the same meaning and period. Some of them are words with different spellings and definitions. The KAROMA method's success criterion is the ability to identify the Karonese term pair that is identical to the actual observed by transcribers. For example, the KAROMA algorithm identifies K^1 if the KAROMA identification result is 'Yes' and the actual observation is also 'Yes'. The rule also applies to another Karonese term pair.

Table 8 *KAROMA evaluation results using member checking*

K	Karonese terms pair	Do Karonese terms exhibit variations in spellings and pronunciations while retaining the same meaning and in time?		Is KAROMA valid?
		Actual observation	KAROMA identified	
1	*'Rimo' - 'Rimau'*	Yes	Yes	Yes
2	*'Lit' – 'Let'*	Yes	Yes	Yes
3	*'Ola' – 'Oula'*	Yes	Yes	Yes
4	*'Aula' – 'Ulah'*	No	No	Yes
5	*'Harimau' – 'Arimau'*	Yes	Yes	Yes
64	*'Kade-Kade' - 'Kade'*	No	Yes	No
100	*'Jong' – 'Jang'*	No	No	Yes

Table 8 presents detailed information on the identification results of the KAROMA algorithm. KAROMA algorithm succeeded in identifying 99 of 100 Karonese input pairs. In other words, the accuracy of the KAROMA algorithm is 99%, using a member checking evaluator. Unfortunately, a Karonese term pair has not been identified and reworded. That is the 64th row ('*kade-kade' - 'kade'*). The term '*kade-kade*' (family) is not plural or a repetition of '*kade*'. The two terms are distinct. It shows that the KAROMA algorithm has difficulty identifying a word that is composed of other words.

4.3 Evaluated KAROMA using Text Similarity-based

This section describes and examines the performance of the KAROMA algorithm based on a text similarity-based approach. This study designed 211 sets of Karonese synthetic sentences and the example shown in Table 5. The success of the KAROMA algorithm as a stemmer and lemmatizer is seen in the text similarity value of Karonese synthetic sentences in *K*. If the text-similarity of the sentences is one, then the accuracy is 100%, and the KAROMA algorithm is declared successful in identifying Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning.

Table 9 shows the KAROMA's performance. KAROMA was successful in identifying 205 of 211 Karonese term sets with 100% accuracy. The rest is KAROMA accuracy < 50%, and with the information that KAROMA is not optimally functioning as a Karonese stemmer and lemmatizer. So, based on the text similarity-based approach, KAROMA has a 97.16% accuracy rate in identifying Karonese terms with different spelling pronunciations but the same meaning.

Table 9 *KAROMA performance by using text similarity-based*

K	Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and in time?	Accuracy (%)	Description
1	*{'rimo','rimau','rimou'}*	100	Success
2	*{'lit',' let'}*	100	Success
3	*{'aula','ula','ola','oula'}*	100	Success
11	*{'langa','langnga','lenga','lengga','lengnga'}*	100	Success
.	.	.	
.	.	.	
.	.	.	
115	***{'gambir', 'gamber'}***	50	Fail
128	***{'mbue', 'bue'}***	50	Fail
131	***{'surung', 'surong'}***	50	Fail
146	***{'ise', 'isei', 'isai'}***	33.33	Fail
195	***{'la' , 'lo'}***	50	Fail
211	***{'ue', 'uei' , 'uai'}***	33.33	Fail

In addition, this session presents a simulation of calculating the success and shortcomings of KAROMA in identifying Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and in time. The KAROMA algorithm successfully identified the Karonese terms of K_3^1. The breakdown of these successes is shown in text similarity of K_3^1 in Table 10. There are three Karonese terms of K_3^1=

J. of Soft Computing and Data Mining Vol. 5 No. 1 (2024) p. 91-103

{*'rimo','rimau','rimou'*}, which means there are three SS, these are SS_1^1, SS_2^1, SS_3^1. The success is determined by *ModCosine* value of text similarity matrix is one. In other words, KAROMA algorithm is 100% successful in identifying the Karonese terms of K_3^1.

Table 10 *Text similarity matrix of* K_3^1

	SS_1^1	SS_2^1	SS_3^1
SS_1^1	1	1	1
SS_2^1	1	1	1
SS_3^1	1	1	1

Opposite, the KAROMA algorithm is not fully successful in detecting Karonese terms of K_3^{146}. There are three Karonese terms of K_3^{146} = {*'isai','ise','isei'*}, which means there are three SS. These are SS_1^{146}, SS_2^{146}, SS_3^{146} KAROMA's imperfections in identifying Karonese terms of K_3^{146} because some *ModCosine* values of SS_1^{146}, SS_2^{146}, SS_3^{146} are zeros. Table 11 presents text similarity matrix of K_3^{146}. It is just diagonal. The matrix is one, another is zero. It means that there are imperfections on Karo WordNet that contain the term. In such a way that KAROMA undetected the Karonese terms. It is also occurred on K_2^{115}, K_2^{128}, K_2^{131}, K_3^{146}, K_2^{195}, *and* K_2^{211}.

Table 11 *Text similarity matrix of* K_3^{146}

	SS_1^{146}	SS_2^{146}	SS_3^{146}
SS_1^{146}	1	0	0
SS_2^{146}	0	1	0
SS_3^{146}	0	0	1

4.4 KAROMA Performance Analysis

In addition, this study also compared the KAROMA algorithm with other approaches. The idea is to use an SS as object testing. There are three approaches that are compared: Word2Vec with cosine similarity, Jaccard similarity, and TF-IDF with cosine similarity is better than existing methods to detect Karonese terms with different spelling pronunciations but the same meaning. This statement is proven by presenting the effect of KMA on measuring Karones text similarity. Table 12 shows that the existing methods do not work well to calculate similarity between two Karonese sentences, which contain Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and in time. Identical Karonese sentences do not have a similarity value of one. In contrast, the proposed work is capable of determining that the sentences are identical.

Table 12 *Comparison of similarity of Karonese sentences using several methods*

Sentences	Word2vec + Cosine	Jaccard Similarity	Cosine Similarity	TF-IDF + Cosine Similarity	**KAROMA + Cosine Similarity**
S_1 = 'ula kam man jaung.', S_2 = 'oula kam man jong.'	0.019	0.33	0.5	0.336	1

5. Conclusion

This research has provided a Karonese morphology analyzer (KAROMA algorithm) to stem and lemmatize Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and in time. The approach taken is word-based morphology, which is combined with the concept of graph theory. The combination produces a Karonese WordNet, which is loaded into the KAROMA algorithm. This study also conceptualizes two algorithms to evaluate KAROMA algorithm performance: member checking and text similarity-based by modified cosine similarity function. KAROMA algorithm successfully identified Karonese terms that exhibit variations in spellings and pronunciations while retaining the same meaning and, in time,

normalized the terms mentioned. This success has an accuracy of 99% by member checking and 97.16% by text similarity-based. Surely, the existence of the KAROMA algorithm as a Karonese morphology analyzer can contribute to the development of NLP research in Karonese. Future work should include the implementation of the KAROMA algorithm for NLP applications such as Karonese sentiment analysis, text summarization, and many more. In addition, the Karonese Stopword library is an important text preprocessing tool in NLP for the Karonese language that can be improved.

Acknowledgement

The authors would like to thank the Faculty of Mathematics and Natural Science, Big data and Artificial Intelligent Research Center Medan State University and Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia (UTHM) for supporting this work.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

Draft and revised manuscript, design, running and analyze experiment: *Ichwanul Muslim Karo Karo;* ***validation of methodology and results interpretation:*** *Mohd Farhan Md Fudzee;* ***paper conception:*** *Shahreen Kasim and Azizul Azhar Ramli. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Imin, G., Ablimit, M., Yilahun, H., & Hamdulla, A. (2022). A Character String-Based Stemming for Morphologically Derivative Languages, *Information (Switzerland)*, *13(4),* https://doi.org/10.3390/info13040170.

[2] Debbarma, K., Patra, B. G., Das, D., & Bandyopadhyay, S. (2012). Morphological Analyzer for Kokborok, *In 3rd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP-2012).*

[3] Urmi, T. T., Jammy, J. J., Ismail, S., Das, S., Mitra, P., Paik, J. H., Parui, S. K., Ganguly, D., Leveling, J., Jones, G., Hoque, M. N., Seddiqui, M. H., Mahmud, M. R., Afrin, M., Razzaque, M. A., Miller, E., Iwashige, J., Sarkar, S. S., Bandyopadhyay, S., … Gupta, V. (2013). Morphological Stemming Cluster Identification for Bangla. In *Proceeding of the 7th International Conference on Computer and Information Technology (ICCIT), Bangladesh, 2004, 5(1)*.

[4] Abdurakhmonova, N., Alisher, I., & Sayfulleyeva, R. (2022). MorphUz: Morphological Analyzer for the Uzbek Language, *7th International Conference on Computer Science and Engineering (UBMK)*, 61–66. https://doi.org/10.1109/UBMK55850.2022.9919579.

[5] Rajasekar, M., & Geetha, A. (2022). Comparison of Machine Learning Methods for Tamil Morphological Analyzer, Lecture Notes in Networks and Systems, 213. https://doi.org/10.1007/978-981-16-2422-3_31.

[6] Sai Kiranmai, G., Mallika, K., Anand Kumar, M., Dhanalakshmi, V., & Soman, K. P. (2010). Morphological Analyzer for Telugu Using Support Vector Machine, *Communications in Computer and Information Science, 101*. https://doi.org/10.1007/978-3-642-15766-0_68.

[7] Abeera, V. P., Aparna, S., Rekha, R. U., Anand Kumar, M., Dhanalakshmi, V., Soman, K. P., & Rajendran, S. (2012). Morphological analyzer for Malayalam using machine learning, Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6411 LNCS. https://doi.org/10.1007/978-3-642-27872-3_38.

[8] Woollams, G. (1996). Pacific Linguistics A Grammar of Karo Batak, Sumatra. https://doi.org/10.15144/PL-C130.cover.

[9] Alajmi, A. F., Saad, E. M., & Awadalla, M. H. (2011). Hidden markov model based Arabic morphological analyzer, *International Journal of Computer Engineering Research, 2(2)*.

[10] Karo Karo, I. M., Fudzee, M. F. M., Kasim, S., & Ramli, A. A. (2022). Sentiment Analysis in Karonese Tweet using Machine Learning. *Indonesian Journal of Electrical Engineering and Informatics, 10(1),* 219–231. https://doi.org/10.52549/ijeei.v10i1.3565.

[11] Karo Karo, I. M., Farhan, M., Fudzee, M., Kasim, S., & Ramli, A. A. (2022). Karonese Sentiment Analysis: A New Dataset and Preliminary Result, *JOIV: International Journal on Informatics Visualization, 6(2–2),* 523–530. www.joiv.org/index.php/joiv.

[12] Muchtar, M., & Waty Kembaren, F. R. (2018). Translation Techniques and Quality in the English Version of Nganting Manuk Text, *International Journal on Language, Research and Education Studies, 2(2),* 195–207. https://doi.org/10.30575/2017/IJLRES-2018050804.

[13] Risnawaty, Sutikno, Sembiring, M., Andriany, L., & Siregar, R. (2020). The translation of Ngangkat Tulan-Tulan texts in karonese society into English, *Talent Development and Excellence, 12(1)*.

[14] Sembiring, M., & Girsang, M. (2019). Translation Procedures of Sijalapen in Karonese Wedding Ceremony Into English, *AICLL: Annual International Conference on Language and Literature, 2(1)*. https://doi.org/10.30743/aicll.v2i1.59.
[15] Wełna, J. (2017). Chapter 4: Morphology. *In Middle English*. https://doi.org/10.1515/9783110525328-004.
[16] Ali, M., Khalid, S., & Aslam, M. H. (2017). Pattern Based Comprehensive Urdu Stemmer and Short Text Classification, *IEEE Access, 6*. https://doi.org/10.1109/ACCESS.2017.2787798.
[17] Ingólfsdóttir, S., Loftsson, H., Daðason, J., & Bjarnadóttir, K. (2019). Nefnir: A high accuracy lemmatizer for Icelandic. https://www.malfong.is/index.php?lang=en&pg=mim.
[18] Wibawa, A. P., Dwiyanto, F. A., Zaeni, I. A. E., Nurrohman, R. K., & Afandi, A. (2020). Stemming javanese affix words using nazief and adriani modifications. *Jurnal Informatika, 14(1)*. https://doi.org/10.26555/jifo.v14i1.a17106.
[19] Jalil, M. M. A., Ismailov, A., Rahim, N. H. A., & Abdullah, Z. (2017). The development of the uzbek stemming algorithm. *Advanced Science Letters, 23(5)*. https://doi.org/10.1166/asl.2017.8332.
[20] 'Osman, O., & 'Mikami, Y. (2012). Stemming Tigrinya Words for Information Retrieval, *Proceedings of COLING 2012: Demonstration Papers,* 345–352.
[21] Tukeyev, U., Turganbayeva, A., Abduali, B., Rakhimova, D., Amirova, D., & Karibayeva, A. (2018). Lexicon-free stemming for Kazakh language information retrieval. *IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018 - Proceedings*. https://doi.org/10.1109/ICAICT.2018.8747021.
[22] Akilan, R., & Naganathan, E. R. (2015). Morphological analyzer for Classical Tamil text: A rule-based approach. *ARPN Journal of Engineering and Applied Sciences, 10(20)*.
[23] Forbes, C., Nicolai, G., & Silfverberg, M. (2021). An FST morphological analyzer for the Gitksan language. *SIGMORPHON 2021 - 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Proceedings of the Workshop*. https://doi.org/10.18653/v1/2021.sigmorphon-1.21.
[24] Shalal, F. A. (2018). Morpheme-based approach versus word-based approach: classifying derivative words with respect to their bases. *Russian Linguistics*, *42*(2). https://doi.org/10.1007/s11185-018-9197-5.
[25] Shah, D. N., & Bhadka, H. (2020). Paradigm-based morphological analyzer for the Gujarati Language. *Advances in Intelligent Systems and Computing, 989*. https://doi.org/10.1007/978-981-13-8618-3_50.
[26] Mokanarangan, T., Pranavan, T., Megala, U., Nilusija, N., Dias, G., Jayasena, S., & Ranathunga, S. (2016). Tamil morphological analyzer using support vector machines. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9612*. https://doi.org/10.1007/978-3-319-41754-7_2.
[27] Asker, L., Argaw, A. A., & Gambäck, B. (2006). Applying Machine Learning to Amharic Text Classification. *Discovery*, *11*(3).
[28] Namly, D., Bouzoubaa, K., El Jihad, A., & Aouragh, S. L. (2020). Improving Arabic Lemmatization Through a Lemmas Database and a Machine-Learning Technique. *In Studies in Computational Intelligence (Vol. 874)*. https://doi.org/10.1007/978-3-030-34614-0_5.
[29] Froud, H., Benslimane, R., Lachkar, A., Lachkar, A., & Alaoui Ouatik, S. (2010). Stemming and similarity measures for Arabic Documents Clustering. *2010 5th International Symposium on I/V Communications and Mobile Networks, ISIVC 2010*. https://doi.org/10.1109/ISVC.2010.5656417.
[30] A. Otair, M. (2013). Comparative Analysis of Arabic Stemming Algorithms. *International Journal of Managing Information Technology*, *5*(2). https://doi.org/10.5121/ijmit.2013.5201.
[31] Habibi, M., & Cahyo, P. W. (2020). Journal Classification Based on Abstract Using Cosine Similarity and Support Vector Machine. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *4*(3). https://doi.org/10.14421/jiska.2020.43-06.