Translation Learning Tool for Local Language to Bahasa Indonesia using Knuth-Morris-Pratt Algorithm

Harco Leslie Hendric Spits Warnars [1], Jessica Aurellia [2], Kendrick Saputra [2]

[1]*Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia*
[2]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia*

Abstract **– During the COVID-19 pandemic time, it is a requirement to deliver online learning since students can not have face-to-face meetings and they depend on gadgets such as a computer, mobile phone, iPad, and laptop to continue their study. One of the subjects is local language learning as this subject is a requirement in some provinces in Indonesia as a gesture concerning local wisdom. However, there is a lack of support for learning the local language since the local languages have a mouth to mouth learning knowledge without any dictionary support. This paper proposed the idea of using the Knuth-Morris-Pratt algorithm to translate the Palembang language to Bahasa Indonesia to help students to learn the Palembang language with the application.**

Keywords **– Knuth Morris Pratt, string matching, translation tool application learning, language-translation tool**

DOI: 10.18421/TEM101-07
https://doi.org/10.18421/TEM101-07

Corresponding author: Harco Leslie Hendric Spits Warnars, *Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia.*
Email: spits.hendric@binus.ac.id

Received: 17 July 2020.
Revised: 04 November 2020.
Accepted: 16 November 2020.
Published: 27 February 2021.

© 2021 Harco Leslie Hendric Spits Warnars, Jessica Aurellia & Kendrick Saputra; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at www.temjournal.com

1. Introduction

Based on data on http://covid19.go.id per 22 May 2020, Indonesia has 20,796 COVID-19 cases and there are 14,413 people in care, 5,057 people recover, and 1,326 dead people. The COVID-19 pandemic directs all worldwide education to use internet technology for delivering the teaching-learning process including Indonesia as well. Everyone was forced to learn how to use technology in any of their activities using meeting applications such as Zoom, Microsoft Teams, Hangout, and many other tool learning activities. However, there are some problems in teaching delivery in some subjects where the transformation knowledge should use the laboratory.

Moreover, some subjects lack technology support, and the knowledge is nurtured by local wisdom knowledge which maintains mouth to mouth delivery of teaching knowledge. Not left behind, teaching the local language is one of the major problems for teaching knowledge delivery, where most of the subject is taught by a local teacher who was born and knows that local language since their childhood, and there is no technology support and lack of government support. Moreover, their knowledge of the local language is obtained because they are part of the local community who care for their local language.

However, not every teacher is equipped with sensible knowledge and want to spend their time and money for the development of the teaching delivery, moreover, to most of the teachers this is as a part-time job when they have others job in other sectors such as agriculture, transportation and so on. These have become the bottleneck of the problems where there is a need for some assistance from the government and teaching society in order to help teachers deliver their teaching subject in the local language.

Indonesia, which is a diverse country, has over 700 regional local languages. One of them is the

Palembang language, frequently used by the people of Palembang in South Sumatera. Some of the Palembang population are newcomers, hence they might experience some difficulty in understanding and using the language [7]. Newcomers who are also students might experience this problem beyond communication, but also in their learning process. Some schools in certain regions require students to learn the local language. Learning local languages can be hard, since it lacks formal learning tools like dictionaries and relying on other people to learn, especially during a situation like this COVID-19 pandemic. One of the most effective ways to learn a new language is through translation, which is currently used in various learning contexts [8].

Meanwhile, string searching is a common task to be performed in this age, either to determine the position of the pattern string sought or to find whether the pattern exists in a text [1]. Its results are applied in various applications, including but not limited to spam filtering, computational biology, and online string matching [2],[3]. There are various string matching algorithms, such as the Knuth-Morris-Pratt (KMP) algorithm. This algorithm was developed separately in 1966 by James H. Morris with Vaughan R. Pratt, and by Donald E. Knuth in 1967. In 1977, it was then published simultaneously by the developers [4]. The KMP algorithm is now one of the most efficient strings matching algorithms in theory, with the time complexity of O(m+n) [5]. Due to its relatively quick execution time when applied to large-sized texts, the KMP algorithm is considered superior to other string matching algorithms [6].

Thus, this research proposes to use the KMP algorithm to solve the aforementioned problem. The KMP algorithm is used to translate words from the Palembang language to Bahasa Indonesia using a sample dictionary. To increase understanding, this research will also provide visualization for the KMP algorithm. The idea then could be implemented in other local languages.

2. Previous and Current Research

2.1. Palembang Language

The Palembang language consists of two dialects, Bebaso and Baso Palembang Seari-ari. Bebaso or Bahasa Palembang Alus is the polite or formal form of the Palembang language, while Baso Palembang Seari-ari is the informal form of the Palembang language. Palembang is a multiethnic society located in South Sumatera, and a notable portion of their society are newcomers from outside of the region [7], [9]. According to Andri (2019), the newcomers in Palembang might neither understand the language nor use it properly [7].

2.2. String Matching and Knuth-Morris-Pratt Algorithm

String matching is an algorithm to search for all occurrences of short strings called patterns in longer strings called text [11]. String matching eases us to check the presence of a pattern in a text. String matching has three kinds of direction to match a pattern string to a text, namely from left to right, from right to left, and from right to left to the second character from the first character to the second character from the last character [12]. String matching consists of various algorithms, specifically the Naive String Matching algorithm, Boyes-Moore algorithm, Rabin Karp Algorithm, and Knuth-Morris-Pratt algorithm (KMP) [13].

Meanwhile, the Knuth-Morris-Pratt algorithm is one of the string matching algorithms, which is developed from the Brute Force Algorithm [12]. The KMP algorithm was discovered separately by James H. Morris, Vaughan R. Pratt, and Donald E. Knuth, who then published the algorithm simultaneously in 1977 [10]. The KMP algorithm is efficient and can be suitable for any string search type [14]. The KMP algorithm uses the prefix and the suffix of the matching. First, we count the failure table of a pattern with the prefix and the suffix. Then, we match the pattern with a text. When a mismatch occurs, the pattern will move according to the number of characters matched previously. This information is found on the failure table [6]. The KMP algorithm is noticeably faster since there is no need for backing up after a partial match [4].

The KMP algorithm has a preprocessing stage where a failure table is generated for each word. The function of the failure table is to identify the maximum possible shift to prevent making unnecessary comparisons, as well as decreasing the time needed to compare both the pattern and text string. Refer to algorithm 1 to make a function to generate a failure table with a time complexity of O(n). Visualization is provided.

Algorithm 1: Algorithm for KMP failure table
Input: pattern string
Output: failure table

Initialization:

```
1: set len = 0, where len is the length
   of the previous longest prefix
   suffix
2: set f[0] to 0
3:   set i to 1
4:while i < length of pattern string do
5:    if pat[i]==pat[len] then
6:          len++
7:               f[i] = len
8:               i++
9:         else
```

```
10:    if len!=0 then
11:         len = f[len-1]
12:    else
13:          f[i]=0
14:            i++
15:         end if
16: end while
```

Tables 1. to 7. are used to illustrate the implementation of Algorithm 1, a preprocessing stage for the Knuth-Morris-Pratt algorithm, which is used to make a failure table. Note that the leftmost column consists of the variables and the asterisk represents the current value of the variable i and len. It also represents the index for the variable prefix function.

Table 1. Pattern string

value/index	0	1	2	3	4	5
Pattern	B	C	D	B	A	A

The failure table made is based on the pattern string found in Table 1., which is BCDBAA. The visualization for the initialization stage is found in Table 2., wherein this stage, len is first set to 0. Then, the value of f[0], i.e., the index 0 of the prefix function, is set to 0. Next, the variable i is set to 0.

Table 2. Initialization

value/index	0	1	2	3	4	5
i		*				
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0					

The result of the first iteration is found in Table 3. Since i is less than the length of the pattern string, the loop is executed. Moreover, pattern[i] is not equal to pattern[len], and len is equal to zero, hence line 13 to 14 of Algorithm 1 is executed. First, set f[i], i.e., the index 1 of prefix function, to 0. Next, increment the variable i. Thus, the value of the variable i is now 2.

Table 3. After 1st iteration

value/index	0	1	2	3	4	5
i			*			
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0	0				

Table 4. shows the result of the second iteration. Since i is still less than the length of the pattern string, the loop is executed again. In this stage, pattern[i] is not equal to pattern[len] and len is equal to zero, therefore line 13 to 14 of Algorithm 1 is executed. First, set f[i], i.e., the index 2 of prefix function, to 0. Next, increment the variable i become 3.

Table 4. After 2nd iteration

value/index	0	1	2	3	4	5
i				*		
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0	0	0			

The result of the third iteration is shown in Table 5. Like the previous steps, the value of the variable i is less than the length of the pattern string, hence, the loop is executed. However, pattern[i] is now equal to pattern[len], therefore line 6 to 8 of Algorithm 1 is executed. First, the variable len is incremented, making its value now equal to 1. f[i] is set to the value of len, meaning the index 3 of the prefix function is set to 1. Finally, the variable i is incremented, making its value 4.

Table 5. After 3rd iteration

value/index	0	1	2	3	4	5
i					*	
len		*				
Pattern	B	C	D	B	A	A
Prefix function	0	0	0	1		

Table 6. shows the result of the fourth iteration. The value of the variable i is still less than the length of the pattern string, hence, the loop is once more executed. Pattern[i] is no longer equal to pattern[len], however, the value of len is no longer equal to 0, hence the 11th line of Algorithm 1 is executed. The variable len is set to f[len-1], meaning len is set to 0.

Table 6. After 4th iteration

value/index	0	1	2	3	4	5
i					*	
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0	0	0	1		

The result of the 5th iteration is shown in Table 7. The loop is executed because the value of the variable i is still less than the length of the pattern string. Since pattern[i] is no longer equal to pattern[len] and the value of len is equal to 0, therefore line 13 to 14 of Algorithm 1 is executed. F[i], the fourth index of the prefix function, is set to 0. Also, the variable i is incremented, making its value 5.

Table 7. After 5th iteration

value/index	0	1	2	3	4	5
i						*
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0	0	0	1	0	

Table 8. shows the result of the 6th iteration. The loop is executed because the value of the variable i is 5, was less than the length of the pattern string, 6. Pattern[i] is not equal to pattern[len] and len is not equal to 0, thus line 13 to 14 of Algorithm 1 is executed again. F[i], which is the fifth index of the prefix function, is set to 0. The variable i is then incremented to 6, which prevents the loop from starting again. With that, the failure table for the pattern "BCDBAA" is completed.

Table 8. After 6th iteration

value/index	0	1	2	3	4	5
i						*
len	*					
Pattern	B	C	D	B	A	A
Prefix function	0	0	0	1	0	0

After generating the failure table, we can use the failure table to match the pattern string to the text. Our research will use this algorithm to match the user input to the words in our dictionary. This section has a time complexity of O(m). Refer to Algorithm 2 to match the pattern string to the text.

Algorithm 2: Algorithm for KMP
Input: pattern string, text string

Initialization:

```
1: int M = length of pattern string
2: int N = length of text string
3: int f[M]
4: int i = 0 //index for text array
5: int j = 0 //index for pattern array
6: while i < length of text string do
7: if pat[j]==txt[i] then
8:          j++
9:          i++
10: end if
11: if j==M then
12:    j = f[j-1]//pattern in location
13: else if i < N and pat[j]!=txt[i]
then
14:      if j!=0 then
15:         j = f[j-1]
16:      else
17:          i++
18:      end if
19: end while
```

Tables 9. - 11. are used to illustrate the implementation of Algorithm 2, the matching stage of the Knuth-Morris-Pratt algorithm, used to compare the pattern string with the text string, which is represented by the variable letter. In this case, the pattern string is "BCDBAA" and the text string is "BBCDBCDBAA". The text string is shown in the first two rows of Table 9. whilst the pattern is shown in the last three rows of Table 9. Thus, Table 9. shows the first comparison pattern string or letter "BBCDBCDBAA" as shown in the first two rows in Table 9. and compared to the pattern "BCDBAA" from left to right as shown in rows 3 and 4 in Table 9. Meanwhile, the last row in Table 9. shows prefix function (0,0,0,1,0,0) from algorithm 1 above, as references from the last row in Table 8.

The comparison starts with the first character of the pattern with the first character of the letter starting from left to right. The comparison starts from the pattern with index=0 (Pattern[0]=B) and compared to the letter with index=0 (Letter[0]=B) refer to as shown in the 2nd column of Table 9. Next, the pattern index=1 (Pattern[1]=C) is compared to the letter index=1 (Letter[1]=B) and they are mismatched as shown in the 3rd column of Table 9. Since the comparison is a mismatch, then the comparison should be changed where the pattern should be moved to the last index mismatch position in the letter and this is the case in index=1 and Table 10. shows the changing comparison position.

Table 9. First comparison

index Letter	0	1	2	3	4	5	6	7	8	9
Letter	B	B	C	D	B	C	D	B	A	A
index Pattern	0	1	2	3	4	5				
Pattern	B	C	D	B	A	A				
Prefix function	0	0	0	1	0	0				

Similar to the first comparison as shown in Table 9., Table 10. shows the comparison between the letter "DBCDBCDBAA" as shown in the first two rows in Table 10. and the pattern "BCDBAA" as shown in rows 3 and 4 in Table 10., but the comparison starts from pattern index 0 (Pattern[0]=B) into letter index 1 (Letter[1]=B). The next three-character comparisons are continued since they have similar comparison letter such as comparison between Pattern[1]=C and Letter[2]=C, Pattern[2]=D and Letter[3]=D, Pattern[3]=B and Letter[4]=B. The comparison between Pattern [4]=A and Letter[5]=C as mismatch comparison, where the character A in pattern mismatch with character C in letter. Because of this mismatch comparison, then the comparison should be changed where the pattern should be moved to the last index mismatch position in the letter and this is the case in index=5 and Table 11. shows the changing comparison position. However, the first position character Pattern[0] is not compared with Letter[5] but compared to Letter[4] because there is prefix function=1 as shown in the last row of Table 10., in index position=4 for the letter (Letter[4]).

Table 10. Second comparison

Array Letter	0	1	2	3	4	5	6	7	8	9
Letter	D	B	C	D	B	C	D	B	A	A
Array Pattern		0	1	2	3	4	5			
Pattern		B	C	D	B	A	A			
Prefix function		0	0	0	1	0	0			

Similar to the first comparison as shown in Table 10., Table 11. shows the comparison between letter "DBCDBCDBAA" as shown in the first two rows in Table 11. and the pattern "BCDBAA" as shown in rows 3 and 4 in Table 11., but the comparison start from Pattern index 0 (Pattern[0]=B) into Letter index 4 (Letter[4]=B). The next five-character comparisons are continued to the next comparison between Pattern[1]=C and Letter[5]=C, Pattern[2]=D and Letter[6]=D, Pattern[3]=B and Letter[7]=B, Pattern[4]=A and Letter[8]=A, Pattern[5]=A and Letter[9]=A show there is no mismatch comparison and this is shows that pattern "BCDBAA" was found in the letter "DBCDBCDBAA" in index position 4 to 9.

Table 11. Third comparison

Array Letter	0	1	2	3	4	5	6	7	8	9
Letter	D	B	C	D	B	C	D	B	A	A
Array Pattern					0	1	2	3	4	5
Pattern					B	C	D	B	A	A
Prefix function					0	0	0	1	0	0

2.3. Applications of the Knuth-Morris Pratt algorithm

String matching is crucial not only in text processing but also in other science fields where patterns need to be found. The applications of string matching algorithms like the Knuth-Morris-Pratt algorithm include, but are not limited to, spell checking, DNA processing, spam filters, and computer vision [15]. This section elaborates on the applications of the KMP algorithm from various science fields.

2.3.1. Learning languages

Sigit and Sulistiyono (2017) discussed the usage of the Knuth-Morris-Pratt algorithm in their learning language application. The application translates the Indonesian words or sentences to the Javanese language in either conversational or polite form. The Knuth-Morris-Pratt algorithm is used to improve the performance of searching Indonesian words that will be translated [9].

Handrizal et al. (2017) analyzed the usage of the Knuth-Morris-Pratt algorithm to create a simple English to English dictionary. If the pattern inputted by the user is found in the dictionary, then the application will display the words that contain the said pattern, as well as the definition of the words. The complexity of the Knuth-Morris-Pratt algorithm is $\theta(m + n)$, where $\theta(m)$ is the preprocessing phase and $\theta(n)$ is the searching phase [16].

2.3.2. Comparing DNA sequence

One of the applications of the Knuth-Morris-Pratt algorithm in the bioinformatics field is to detect unusual patterns in the human DNA which may raise the risk of health problems for that particular human. The disease patterns are then matched with the genome database sequences. Considering that the genome databases are large, to match the disease pattern is a complex process. Thus, the KMP algorithm is used to match patterns in less time with higher accuracy [17].

2.3.3. Detecting malicious or spam emails

Sesha Rao et al. (2017) used the Knuth-Morris-Pratt algorithm in their system model, which is used to detect malicious or spam emails, specifically in the data processing phase. The KMP algorithm is used to identify the occurrences of words from the mail body. These data are then analyzed and classified using principal component analysis and data mining techniques. Then, it could be used to detect and stop spam senders in the future [18].

2.3.4. Spell checker

The Knuth-Morris-Pratt algorithm is used, along with a corpus, to make a spell checker for the Somali language. The KMP algorithm is used in the processing stage to produce suggestion lists for the misspelled words [19]. This algorithm increases the speed of pattern matching between misspelled words and Somali words found in the corpus. If no matches are found, the application will mark the spelling error, and then it will offer suggestions that have the same structure as the misspelled word [22].

N-grams and the Knuth-Morris-Pratt algorithm are combined to create a Chinese spelling checker with a Cantonese correction system [20]. First, the N-gram is used to detect the probability of a sentence formed by the writers, while the KMP algorithm is used to find and correct the mistakes. The KMP algorithm is used to look for a particular string in a sentence and to note the index of the target string. When a word from the dictionary appears in sentences, the system will note the translation of Cantonese usage as well as the position of the word [23].

2.3.5. Intrusion detection system

The Knuth-Morris-Pratt algorithm was used to detect intruders in wireless sensor networks (WSN) [21]. The National Institute of Standards and Technology defines WSN as a network that consists of many wireless sensors that are used both in a geographical area and in the environment. Today, it

is commonly used in both military and civil jobs [24]. However, it faces a notable problem in security, namely the WSN is unprotected in an open environment. The KMP algorithm checks if the given sequences of packet tokens match with the malicious codes stored in a database. The system will then give a warning of intrusive behavior when a match is found [25].

3. Proposed Idea

One of the applications of the Knuth-Morris-Pratt algorithm is to make a program that can assist people in learning languages. To help the newcomers in Palembang to understand the language better, a Palembang-Bahasa Indonesia dictionary could be created.

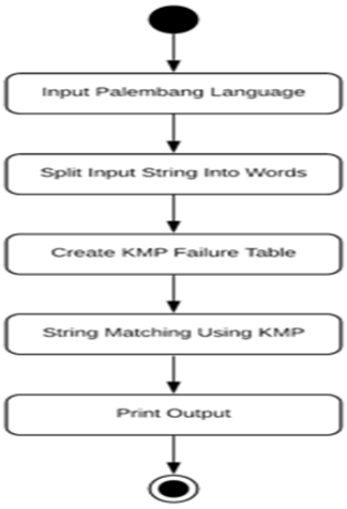

Figure 1. Knuth Morris Pratt algorithm steps

First, the input sentence is broken down by words. Each word acts as the pattern string. Then, the Knuth-Morris-Pratt algorithm is used to match the words inputted by the user to the one in the dictionary. If a match is found, the program will return the Bahasa Indonesia translation as the output. The following activity diagram in Figure 1. describes the algorithm of our program.

To better visualize the way the proposed program works, examples can be found below. In addition, an index is added to explain the algorithm better. From this point on, the following sample dictionary found in Table 12. will be used to illustrate the mechanisms of our proposed idea, wherein Table 12. there are 3 columns. In the first column as an index searching, the second column contains the Palembang word and the third column as Indonesian word translation for each Palembang word in the second column. In this case, we use 17 words translation between the Palembang language and the Indonesian language. Some words have a similar meaning, for example as Palembang language word "minum" in index 10 has Indonesian word translation "minum".

Table 12. Dictionary table

index	Palembang	Indonesian
0	abah	ayah
1	aku	saya
2	apo	apa
3	banyu	air
4	bulet	bulat
5	cakmano	bagaimana
6	galo	semua
7	idak	tidak
8	iwak	ikan
9	mak	ibu
10	minum	minum
11	manusio	manusia
12	mato	mata
13	palak	kepala
14	tegak	berdiri
15	ujan	hujan
16	wong	orang

For example, we execute the algorithm with 3 different inputs. Firstly, we use the input statement: "mato abah idak bulet" as Palembang language and the output is "mata ayah tidak bulat" as Indonesian language translation and Figure 2. shows the print screen of the application. By using the Knuth-Morris-Pratt algorithm, the first word of the input string "mato" is matched to the word in index 12 of the sample dictionary. Hence, the program will return the word *"mata"* as the first-word translation of the input. The second word of the input "abah" is matched to the word in index 0 of the sample dictionary. Thus, the output of the second word translation is "*ayah"*. The same process can be done for the third and the fourth input words such as "idak" and "bulet" which match the word in index 7 and 4 and returns the word translation "*tidak"* and "*bulat"* respectively.

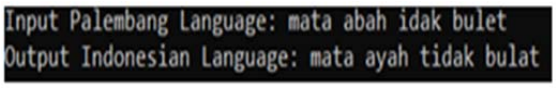

Figure 2. Program result for input 1

The same as the previous example, the second example has input Palembang language statement: "aku minum banyu ujan" and the output is "saya minum air hujan" as Indonesian language translation and Figure 3. shows the print screen of the application. The same as before, by using the Knuth-Morris-Pratt algorithm, the first word of the input string "aku" is matched to the word in index 1 of the sample dictionary and has Indonesian language word translation "saya". The same process has been done on the next Palembang language word such as "minum","banyu","ujan" which match the word in index 10,3,15 and have Indonesian language word translation such as "minum", "air", "hujan" respectively. As mentioned before where there is word similarity between the Palembang language and Indonesian language such as Palembang language word "minum" in index 3 has a similar Indonesian language word translation "minum".

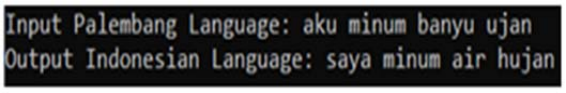

Figure 3. Program result for input 2

The third example as shown in Figure 4., the input Palembang language statement: “palak mak aku cakmano” and the output is “kepala ibu saya bagaimana” as the Indonesian language translation. The Palembang language words such as “palak”, ”mak”, ”aku”, ”cakmano” which match the words in index 13,9,1,5 and have Indonesian language word translation “kepala”, ”ibu”, ”saya”, ”bagaimana” respectively.

Figure 4. Program result for input 3

4. Conclusion

As one of the most efficient strings matching algorithms, the Knuth-Morris-Pratt algorithm is currently applied in many scientific fields. As proposed in this paper, the Knuth-Morris-Pratt algorithm is implemented in the linguistic field to create a program to translate sentences in the Palembang language to Bahasa Indonesia. The algorithm of this program is to first break the input sentence down by words which will act as the pattern string. To continue with the string matching process, the program generates a failure table for each word. Next, the words are matched with ones in the dictionary using the Knuth-Morris-Pratt algorithm. Then, the program will return the Bahasa Indonesia translation as the output when a match is found between the input and the words in the dictionary. The complexity of the algorithm is O(m+n).

This idea can then be implemented in other local languages to help newcomers to understand the local languages better. For further development, this application will be extended to the grammar syntax using semantic meaning such as Natural Language Processing (NLP) which translates not only per word but per statement which includes differentiating each word as a subject, predicate or verb, object such as noun, adjective, and adverb.

Hopefully, creating this local language translator will help teachers and students to learn and adapt the local language easily beyond the background knowledge expertise of teachers regarding the area of the local language used. Moreover, it will help the government and for those who are interested to preserve their local language and transform it into the next generation where the local language can be easily understood by everyone interested to learn it.

References

[1]. Dalal, N. R., & Jadhav, P. (2015). A Composite Algorithm for String Matching. *International Journal of Modern Trends in Engineering and Research (IJMTER)*, *2*(7), 68-73.

[2]. Pandey, G., Martolia, M., & Arora, N. (2017). A Novel String Matching Algorithm and Comparison with KMP Algorithm. *International Journal of Computer Applications*, *179*(3), 6-8.

[3]. Tsarev, R. Y., Chernigovskiy, A. S., Tsareva, E. A., Brezitskaya, V. V., Nikiforov, A. Y., & Smirnov, N. A. (2016, April). Combined string searching algorithm based on knuth-morris-pratt and boyer-moore algorithms. In *IOP conference series: materials science and engineering* (Vol. 122, No. 1, p. 012034). IOP Publishing.

[4]. Knuth, D. E., Morris, Jr, J. H., & Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM journal on computing*, *6*(2), 323-350.

[5]. Lu, X. (2019, November). The Analysis of KMP Algorithm and its Optimization. In *Journal of Physics: Conference Series* (Vol. 1345, No. 4, p. 042005). IOP Publishing.

[6]. Park, N., Park, S., & Lee, M. (2020). High performance parallel KMP algorithm on a heterogeneous architecture. *Cluster Computing*, *23*(3), 2205-2217.

[7]. Andri, A. (2019). Penerapan Algoritma Pencarian Binary Search dan QuickSort pada Aplikasi Kamus Bahasa Palembang Berbasis Web. *Jurnal Informatika: Jurnal Pengembangan IT*, *4*(1), 70-74.

[8]. Karimah, A. (2020). Revisiting Translation As A Foreign Language Learning Tool: Contrasting Beliefs Of Diversely Proficient Students. *SAGA: Journal of English Language Teaching and Applied Linguistics*, *1*(1), 9-16.

[9]. Nadya, N. L. (2018, July). Penggunaan Dan Makna Kata" Gawe" Terhadap Kebiasaan Masyarakat Palembang. In *Prosiding Seminar Nasional Program Pascasarjana Universitas Pgri Palembang* (Vol. 5, No. 05).

[10]. Sigit, H. T. (2017). Application For Learning Javanese Language Of Banten Using Knutt Morris Pratt Algorithm. *International Journal of Advanced Research in Computer Science*, *8*(8), 495.

[11]. Rusito, R., & Khasanah, N. (2019). Aplikasi Pencarian Dengan Menggunakan Algoritma Knuth Morris Pratt Pada Berkas Dokumen Shipment. *JURIKOM (Jurnal Riset Komputer)*, *6*(3), 245-254.

[12]. Novianti, N., Kembaren, R. C. G. I., Bangun, D. M. B., & Marbun, N. (2019). Implementasi Algoritma Knuth Morris Pratt Pada Aplikasi Sinopsis Film Bioskop Berbasis Web. *Komik (Konferensi Nasional Teknologi Informasi dan Komputer)*, *3*(1).

[13]. Ependi, U., & Oktaviani, N. (2017). Abstract Keyword Searching with Knuth Morris Pratt Algorithm. *Scientific Journal of Informatics*, *4*(2), 150-157.

[14]. Janani, R., & Vijayarani, S. (2016). An efficient text pattern matching algorithm for retrieving information from desktop. *Indian Journal of Science and Technology*, *9*(43), 1-11.

[15]. Baig, M. B., & Li, T. S. (2018, October). Parallel String Matching for Urdu Language Text. In *International Conference on Intelligent Technologies and Applications* (pp. 369-378). Springer, Singapore.

[16]. Handrizal, H., Budiman, A., & Ardani, D. R. (2017). Implementation and Analysis Zhu-Takaoka Algorithm and Knuth-Morris-Pratt Algorithm for Dictionary of Computer Application Based on Android. *IJISTECH (International Journal of Information System & Technology)*, *1*(1), 8-21.

[17]. Rahate, P. M., & Chandak, M. B. (2018). Comparative Study of String Matching Algorithms for DNA dataset. *International Journal of Computer Sciences and Engineering*, *6*(5), 1067-1073.

[18]. Rao, A. S., Avadhani, P. S., & Chaudhuri, N. B. (2017). Detecting Targeted Malicious E-Mail Using Linear Regression Algorithm with Data Mining Techniques. In *Computational Intelligence in Data Mining* (pp. 23-35). Springer, Singapore.

[19]. Jimale, A. O., Zainon, W. M. N. W., & Abdullahi, L. F. (2018, June). Spell Checker for Somali Language Using Knuth-Morris-Pratt String Matching Algorithm. In *International Conference of Reliable Information and Communication Technology* (pp. 249-256). Springer, Cham.

[20]. Yeh, J. F., Chang, L. T., Liu, C. Y., & Hsu, T. W. (2017, December). Chinese spelling check based on N-gram and string matching algorithm. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)* (pp. 35-38).

[21]. Kalnoor, G., & Agarkhed, J. (2018). Detection of intruder using KMP pattern matching technique in wireless sensor networks. *Procedia Computer Science*, *125*, 187-193.

[22]. Chia, K. S., & Yap, X. Y. (2016). A portable PID control learning tool by means of a mobile robot. *International Journal of Online and Biomedical Engineering (iJOE)*, *12*(06), 54-57.

[23]. Al-Maroof, R. S., Salloum, S. A., AlHamadand, A. Q., & Shaalan, K. (2020). Understanding an Extension Technology Acceptance Model of Google Translation: A Multi-Cultural Study in United Arab Emirates. *International Journal of Interactive Mobile Technologies (iJIM)*, *14*(03), 157-178.

[24]. Al-Zoubi, A. Y. (2019). Student Active Learning Tool for Producing Open Resources in Microwave Engineering Education. *International Journal of Engineering Pedagogy (iJEP)*, *9*(4), 86-101.

[25]. Thompson, K. S. (2020). Synergetic Learning Model: The Sum is Greater. *International Journal of Advanced Corporate Learning (iJAC)*, *13*(1), 62-76.