Development of A Bilingual Dictionary of Sahu Tala'i – Indonesia Using The Aho-Corasick Algorithm

Pembangunan Kamus Bilingual Bahasa Sahu Tala'i - Indonesia Menggunakan Algoritma Aho-Corasick

Liza Wikarsa[1)], Thomas Suwanto[2)], Christ Henry Loha[3)]

[1,2,3] Program Studi Teknik Informatika Fakultas Teknik Universitas Katolik De La Salle Manado
Kairagi I Kombos Manado, Belakang Wenang Permai II, Manado, 95000, Indonesia

lwikarsa@unikadelasalle.ac.id[1)], tsuwanto@unikadelasalle.ac.id[2)], cristloha2205@gmail.com[3)]

Received : 3 October 2023 || Revised : 20 October 2023 || Accepted: 14 March 2024

Abstract –Nowadays, local languages gain less interest from the younger generation. This also happened to the Sahu Tala'i language which is a regional language originating from a tribe in the Sahu sub-district, West Halmahera district. Hence, users of this language are becoming fewer and will be threatened with extinction if not being consistently used. To preserve the language, this study built an Android-based bilingual dictionary application, for Sahu Tala'i and Indonesian languages, to provide access to the language for people in the Sahu sub-district and those who are interested in learning the language. This study employed the Aho-Corasick algorithm to accelerate word search by matching the strings used to determine the position of certain strings in a larger collection of strings. The test results show that the implementation of the Aho-Corasick algorithm is successfully done to do a word searching in the dictionary. Additionally, an audio feature to pronounce the word in the Sahu Tala'i accent/dialect and examples of the use of these words in sentences are provided in this bilingual dictionary.

Keywords: Bilingual Language, Dictionary, Sahu Tala’i, Aho-Corasick Algorithm.

Abstrak *–Saat ini, bahasa daerah kurang diminati oleh generasi muda. Hal ini juga terjadi pada bahasa Sahu Tala'i yang merupakan bahasa daerah yang berasal dari sebuah suku di Kecamatan Sahu, Kabupaten Halmahera Barat. Oleh karena itu, pengguna bahasa ini semakin sedikit dan akan terancam punah jika tidak digunakan secara konsisten. Untuk melestarikan bahasa tersebut, penelitian ini membangun sebuah aplikasi kamus dwibahasa, Sahu Tala'i dan Indonesia berbasis Android, untuk memberikan akses bahasa tersebut kepada masyarakat di kecamatan Sahu dan mereka yang tertarik untuk mempelajari bahasa tersebut. Penelitian ini menggunakan algoritma Aho-Corasick untuk mempercepat pencarian kata dengan mencocokkan string yang digunakan untuk menentukan posisi string tertentu dalam kumpulan string yang lebih besar. Hasil pengujian menunjukkan bahwa implementasi algoritma Aho-Corasick berhasil dilakukan untuk melakukan pencarian kata dalam kamus. Selain itu, fitur audio untuk mengucapkan kata dalam logat/dialek Sahu Tala'i dan contoh penggunaan kata tersebut dalam kalimat juga disediakan dalam kamus dwibahasa ini.*

Kata Kunci: *Dwibahasa**,** Kamus, Sahu Tala'i, Algoritma Aho-Corasick.*

INTRODUCTION

Indonesia is a country rich in culture and has more than 700 regional languages (Humas Sekretariat Kabinet Republik Indonesia 2023). Regional languages represent regions and tribes to communicate, one of which is the Sahu Tala'i language. This language comes from the Sahu tribe in West Halmahera district. The Sahu Tala'i language is very rarely used by the population there because the younger generation is more inclined to use Indonesian and other foreign languages (Bangowa et al. 2023). This results in fewer Sahu Tala'i language users and will be endangered if not preserved.

To preserve the Sahu Tala'i language, the creation of a bilingual dictionary application, Sahu Tala'i and Indonesian, is considered important. The dictionary contains an alphabetized list of words that includes pronunciation, parts of speech, meaning, history, and word usage (Setiawati 2016). In this study, the bilingual dictionary application will be built using the Aho-Corasick algorithm due to its speed in searching strings with many patterns (Chandra et al. 2020). This string-matching algorithm can also determine the position of a particular string in a larger set of strings or the form of text (Situmorang, Sembiring, and Limbong 2018).

Several studies demonstrated the ability of the Aho-Corasick algorithm to search strings of many patterns quickly (Guo et al. 2023; Chandra et al. 2020; Situmorang, Sembiring, and Limbong 2018; Ginting and Hatmi 2018). Guo et al. (2023) developed an automatic speech recognition (ASR) system capable of identifying previously unseen words, particularly named entities, even if they were not part of the training data. To achieve this, the researchers employed the Aho-Corasick algorithm, which minimized matching errors and significantly enhanced decoding speed and efficiency. The research successfully enhanced Wenet's original contextual biasing method, reducing the number of matching errors in the context graph while improving matching efficiency. Another study, conducted by Chandra et al. (2020), built a custom keyboard application to block porn words using the Aho-Corasick algorithm. The usability test showed that the NFKeyboard application was acceptable and managed to achieve grade B (good value) and excellent adjective ratings. Meanwhile, Situmorang, Sembiring, and Limbong (2018) designed a digital archive application for searching incoming and outgoing mail data at the Poltekkes Kemenkes RI Medan using the Aho-Corasick algorithm. The search feature created was only based on certain categories. Lastly, the studies did by Ginting and Hatmi (2018) focused on implementing the Aho-Corasick algorithm in search of the meaning of slang terms. The results show that this algorithm can provide faster and more accurate search results.

There are differences between those studies and this study which this study categorizes the words into 9 categories and also provides additional features to improve user experience while using the bilingual dictionary of Sahu Tala’i and Indonesia, like pronunciation of words using female and male voices and examples of word usage in sentences. This study aims to implement the Aho-Corasick algorithm in the Android-based Sahu Tala'i-Indonesian bilingual dictionary application that can provide knowledge about this particular language for people who live in the Sahu tribe as well as those who are interested in learning this language for various reasons. Also, this dictionary application is expected to help maintain and document the Sahu Tala'i language so its existence can be further preserved.

This paper is divided into four parts that are introduction, research methodology, results and discussion, as well as conclusions and recommendations. First, the introduction outlines the research background to provide better context by which this study was conducted. Subsequently, it presents the research methodology that explains how the study was done. Then, the results and discussion are delivered before the conclusion and future works.

RESEARCH METHOD

Data Source

The research data were gathered from interviews with 5 respondents from the traditional shop, community, and Balisoan village government who live in the Sahu sub-district. Also, data was collected using a questionnaire created on Google Forms and answered by 336 respondents living in the Sahu sub-district.

Testing of Research Instruments

The level of significance used in this study is 5%. There are two different tests performed on the research instruments, in this case questionnaires, such as validity and reliability tests. This study used SPSS 20 to obtain the results of validity and reliability tests for 13 questions in the questionnaire. There were 336 respondents for this research. Using the formula Df = N-2, therefore Df=334 (Amin, Garancang, and Abunawas 2023).

Validity Test

The validity test is carried out to ensure the validity or suitability of the questionnaire used in obtaining data from respondents (Slamet and Wahyuningsih 2022; Sulistiyowati and Astuti 2021). This study uses the Pearson Correlation validity test, the results of which can be seen in Table 1.

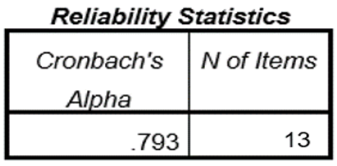
Table 1 Validity Test Results

Item	r-Count	r-Table	Probability Value	Validity
Q1	0,265	0,104552	0,05	Valid
Q2	0,709	0,104552	0,05	Valid
Q3	0,714	0,104552	0,05	Valid
Q4	0,717	0,104552	0,05	Valid
Q5	0,611	0,104552	0,05	Valid
Q6	0,606	0,104552	0,05	Valid
Q7	0,214	0,104552	0,05	Valid
Q8	0,487	0,104552	0,05	Valid
Q9	0,583	0,104552	0,05	Valid
Q10	0,502	0,104552	0,05	Valid
Q11	0,521	0,104552	0,05	Valid
Q12	0529	0,104552	0,05	Valid
Q13	1,000	0,104552	0,05	Valid

Since the r-Count value of each item is greater than the r-Table of 0.104552, then as the basis for decision-making in the validity test it can be concluded that items Q1-Q13 are declared valid.

Reliability Test

The reliability test aims to determine the level of consistency of a questionnaire used so that the questionnaire can be relied upon to measure research variables even though this research is carried out repeatedly with the same questionnaire (Slamet and Wahyuningsih 2022; Sulistiyowati and Astuti 2021).



Reliability Statistics

Cronbach's Alpha	N of Items
.793	13

Figure 1 Reliability Test Results

Based on the Reliability Statistics output in Figure 1, it is known that Cronbach's Alpha value is 0,793. This value will then be compared with the r-Table value with a value of N = 13 sought in the distribution of r-Table values at 5% significance, then the r-Table value is 0,553. Since the Cronbach's Alpha value is 0,793 > 0,104552 (r-Table), it can be concluded that this questionnaire is declared reliable as a data collection tool in this study.

Table 2 below provides an overview of the statistical values for the 13 questionnaire questions. In the "Cronbach's Alpha if Item Deleted" column in this table, the Cronbach's Alpha value for all 13 questions is > 0,60, so it can be concluded that all questionnaire questions are reliable.

Table 2 Reliability Test Results

Item	*Cronbach's If Item Deleted*	r-Table	Reliability
Q1	0,794	0,104552	Reliable
Q2	0,776	0,104552	Reliable
Q3	0,778	0,104552	Reliable
Q4	0,778	0,104552	Reliable
Q5	0,782	0,104552	Reliable
Q6	0,782	0,104552	Reliable
Q7	0,796	0,104552	Reliable
Q8	0,788	0,104552	Reliable
Q9	0,786	0,104552	Reliable
Q10	0,788	0,104552	Reliable
Q11	0,787	0,104552	Reliable
Q12	0,789	0,104552	Reliable
Q13	0,717	0,104552	Reliable

Steps In Aho-Corasick Algorithm

The Aho-Corasick algorithm consists of two parts, the first part serves as a pattern-matching engine for a set of keywords, while the second part applies text strings as input to the pattern-matching engine (Ginting and Hatmi 2018; Jeyaseeli, Mary, and Shanthi 2023). This algorithm scans each character in the text sequentially, without making any jumps or skips. The memory requirement of the Aho-Corasick algorithm can be quite large depending on the length of the pattern (Loukides and Pissis 2022). The pattern-matching process is performed by stepping through the input patterns one by one and selecting the existing matches (Cormen et al. 2022; Kini et al. 2022). Each step in the matching engine occurs in constant time. Therefore, Aho-Corasick matching always runs in O(n) running time which is independent of the number of patterns (Sahara 2014; Kanda, Akabe and Oda 2023).

There are several variables used in the application of the Aho-Corasick method (Chandra et al. 2020; Situmorang et al. 2018; Guo et al. 2023):

Input : n = *input* teks
Process : m *array*[] = {“p1”, “p2”, “p3”,”...”,”pn”}
p1 = *pattern* 1
p2 = *pattern* 2
p3 = *pattern* 3
pn = *pattern* n

Steps in implementing the Aho-Corasick method are as follows (Chandra et al., 2020; Situmorang et al., 2018; Lindblad 2023):

1. Construct the tree and pattern that has been created.
2. Generate a transition table from automata.
3. Do string matching and failure function checking.
4. Display output function.

SOFTWARE DEVELOPMENT METHODOLOGY

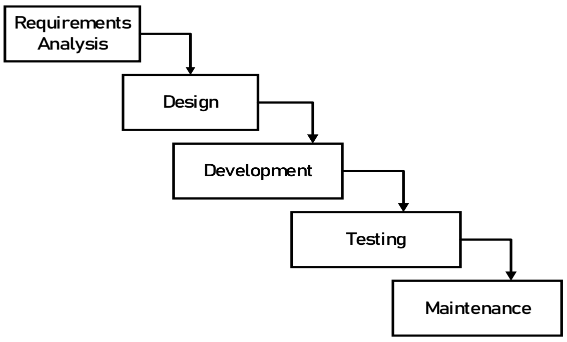


Figure 2 Methodology of Waterfall (Pratama 2021; Budiarto 2020)

The five phases of the waterfall methodology use a sequential development process throughout the project. Each phase has its objectives to accomplish which are as follows (Pratama 2021; Sommerville 2016):

a. 1st phase: requirements analysis
In this first phase, problem identification, target user analysis, requirements specification definition, and others will be carried out.

b. 2nd phase: design
In the second phase, system design will be carried out, including process design, database design, application interface design, and program module design.

c. 3rd phase: development
In the third phase, all the designs that have been done will be implemented with attention to software quality.

d. 4th phase: testing
In this phase, a series of tests will be conducted to ensure that all requirement specifications have been implemented properly and there are no errors in the application.

e. 5th phase: maintenance
In the last phase, there are numerous activities involved such as fixing flaws in the code, design, and algorithm, performance improvement, security enhancement, and others.

This research would not discuss the maintenance phase as the research stopped at the testing phase.

DESIGN

Data Dictionary

Table 3 enlists attribute names, contents, and data types for table Kamus.

Table 3 Data Dictionary for The Bilingual Dictionary Application

Table Name	Attribute Name	Contents	Type
Kamus	audioUrlPria	Contains the URL link to save the male audio	String
	audioUrlWanita	Contains the URL link to save the female audio	String
	contohKataIndo	Contains example sentences from Indonesian	String
	contohKataSahu	Contains example sentences from the Sahu Tala'i language	String
	kataIndonesia	Contains Indonesian vocabulary	String
	kataSahu	Contains Sahu Tala'i language vocabulary	String

Module Program Design

Figure 3 demonstrates how the Aho-Corasick algorithm would be implemented into the bilingual dictionary of Sahu Tala'i and Indonesian. The user would enter keywords into the provided textbox. Next, the algorithm would determine a tree based on the patterns recognized to build a transition table. Once the transition table was created from automata, the algorithm would do string matching and failure function checking. When no matching pattern was found, the failure function would search in each tree. This was done to ensure that no patterns were missed in the string-matching phase. The next step was to perform a vocabulary search in the database. Lastly, it would display the search results back to the user.

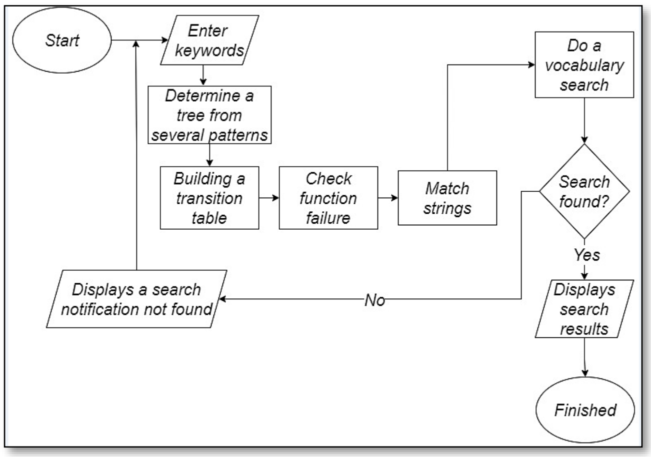


Figure 3 Module For Implementing the Aho-Corasick Algorithm

IMPLEMENTATION

Implementation Environment

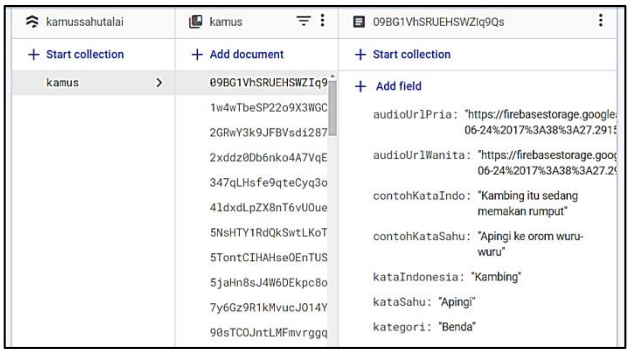
Table 4 contains the list of hardware and software required to develop the bilingual dictionary.

Table 4 Hardware and Software

a. Hardware	
Processor	AMD Athlon 300U
RAM	8 GB
Display device	AMD Radeon(TM) Vega 3 Graphics Primary/Integrated
SSD (Solid State Drive)	500GB
b. Software	
Operating System	Windows 11 Home Single Language
Text Editor	Visual Studio Code 1.79.2
Application Design	Figma 2023
Modeling Tool	Visio Professional 2021
Database	Firebase Firestore
Programming Language	Dart 3.0.5
Minimum Android Version	Android 4.4 KitKat

Database Implementation

Figure 4 displays the dictionary database containing 8 fields that are *audioUrlPria*, *audioUrlWanita*, *contohKataSahu*, *kataSahu*, *contohKataIndo*, *kataIndonesia*, and kategori.



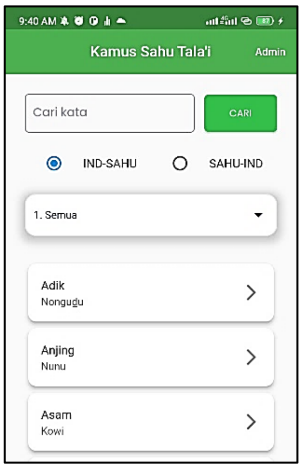
```
09BG1VhSRUEHSWZIq9
1w4wTbeSP22o9X3WGC
2GRwY3k9JFBVsdi287
2xddz0Db6nko4A7VqE
347qLHsfe9qteCyq3o
41dxdLpZX8nT6vU0ue
5NsHTY1RdQkSwtLKoT
5TontCIHAHse0EnTUS
5jaHn8sJ4W6DEkpc8o
7y6Gz9R1kMvucJ014Y
90sTCOJntLMFmvrggq
```

```
audioUrlPria: "https://firebasestorage.google
               06-24%2017%3A38%3A27.2915
audioUrlWanita: "https://firebasestorage.goog
                 06-24%2017%3A38%3A27.29
contohKataIndo: "Kambing itu sedang
                 memakan rumput"
contohKataSahu: "Apingi ke orom wuru-
                 wuru"
kataIndonesia: "Kambing"
kataSahu: "Apingi"
kategori: "Benda"
```

Figure 4 The Database Implementation

Implementation of the Application Interface

Figure 5 is the home page that has a language option and word categories. Once the user inserts the keywords, this user has to select the language option while the word categories are optional as displayed in Figure 6.



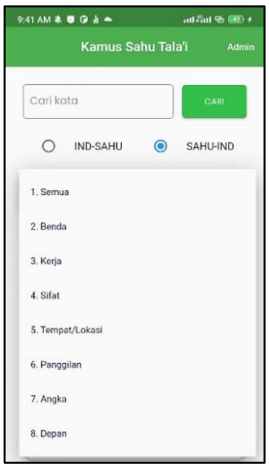
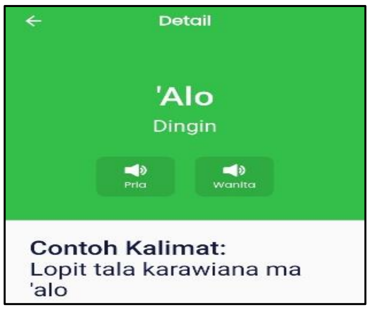


Figure 5 Front Page of the Dictionary Application

Figure 6 The Word Categories

Figure 7 illustrates the search result of the inserted word "alo" and the language option chosen was Sahu Tala'i. The meaning of "alo" is cold. This page provided an example of the use of "alo" in a sentence. Furthermore, the user can also hear the pronunciation of the word by selecting either male or female audio.



Contoh Kalimat:
Lopit tala karawiana ma 'alo

Figure 7 The Word Categories

This dictionary application caters to features for admin that include the addition of new words, update of existing word details, and account management. Figure 8 is the login page for the admin to enter the application.

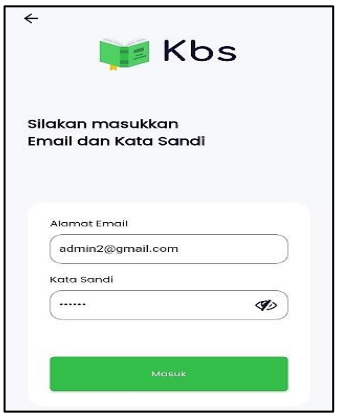
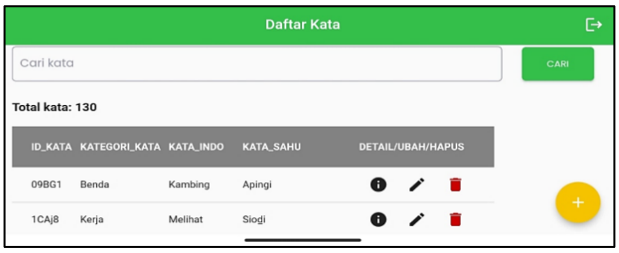


Figure 8 The Login Page

Figure 9 allows the dictionary administrator to search, add, change, and delete data.



Total kata: 130

ID_KATA	KATEGORI_KATA	KATA_INDO	KATA_SAHU	DETAIL/UBAH/HAPUS
09BG1	Benda	Kambing	Apingi	
1CAj8	Kerja	Melihat	Siodi	

Figure 9 The Front Page For Admin

Figure 10 demonstrates the page for adding new words. The file extensions for the male and female audios are mp3, wav, m4a, aac, and ogg with a maximum file size of 500 kb.

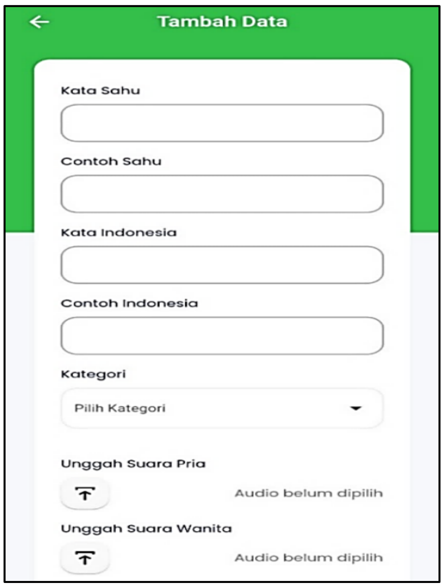


Figure 10 Word Addition

TESTING

Search by Keywords

To search by keywords, the user must insert the keyword into the textbox. For instance, the user inserts three (3) words "Makan Nasa Goreng" as shown in Figure 11. The search results enlist the translation for the words "makan" dan "goreng". It cannot find the translation for the word "nasa" as this word does not exist in the dictionary. Also, this application does not translate the keywords into a single sentence but rather breaks these keywords into separate words. It is noted that this application can only search a maximum of 3 words at once.

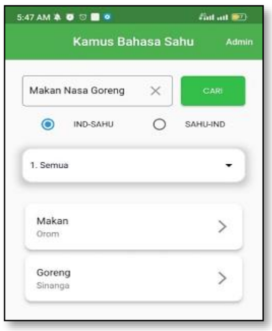


Figure 11 Search By

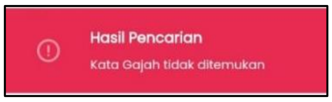


Figure 12 Search Result "Keywords Not Found"

A notification will automatically appear should the keywords not be found as demonstrated in Figure 12.

Search by Word Categories

Aside from searching by keywords, the word categories can be used to find the word(s). There are 8 categories to select from which each category has a collection of related words as can be seen in Figure 13.

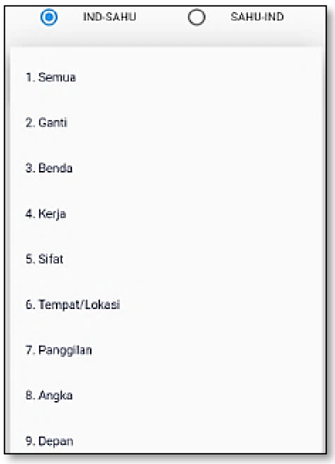


Figure 13 Word Categories

Table 5 enlists the number of words provided for each category in the dictionary.

Table 5 Words in the Dictionary

Category	Total
Ganti (pronoun)	18
Benda (noun)	74
Kerja (verb)	30
Sifat (adjective)	33
Tempat/lokasi (place/location)	10
Panggilan (address)	9
Angka (number)	24
Depan (preposition)	7
Total Kata (word count):	**209**

Adding a New Word

The administrator can add new words along with their details as shown in Figure 14. All fields in this form are mandatory. As for the audio, the application allows to addition of just one audio file that contains the pronunciation of the particular word (s). It also checks the audio file format and file size.

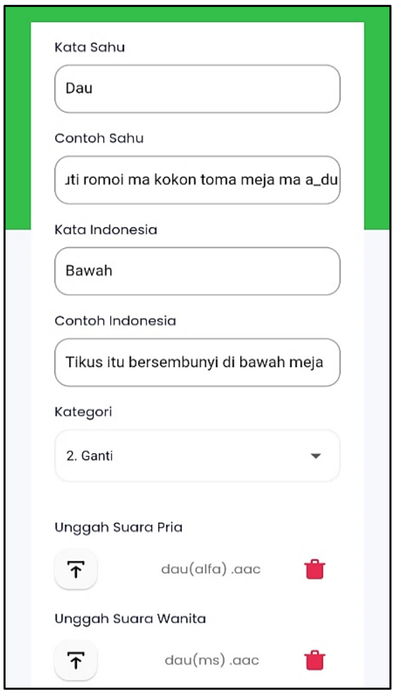

Kata Sahu
Dau
Contoh Sahu
ɹti romoi ma kokon toma meja ma a_du
Kata Indonesia
Bawah
Contoh Indonesia
Tikus itu bersembunyi di bawah meja
Kategori
2. Ganti
Unggah Suara Pria
dau(alfa) .aac
Unggah Suara Wanita
dau(ms) .aac

Figure 14 Adding a New Word

Analysis of Test Results

The followings are the test results of the functionalities provided in the application.

1. This offline dictionary can be run on an Android-based mobile phone with a minimum version of Android 4.4 KitKat.
2. The user can look up words from Sahu Tala'i to Indonesian, and vice versa.
3. The user can only search a maximum of 3 words per search. If it exceeds the word limitation, the search result will not be accurate as there is a need to add more rules to the algorithm.
4. The Aho-Corasick algorithm scans each character in the text sequentially, without making any jumps or skips.
5. The application can provide examples of word usage in sentences.
6. The audio feature for word pronunciation the male and female speakers can be heard clearly.
7. The running time of the Aho-Corasick algorithm is independent of the number of patterns in the text.
8. The greater the number of words in the database, the higher the word search accuracy.

RESULT AND DISCUSSION

Analysis

Problems Identification

From the interviews conducted, the respondents pointed out that the Sahu Tala’i language is very rarely used by the population there. Sahu Tala'i language users only exist in 11 villages where the majority of the population is over 40 years old. While the younger generation is more inclined to use Indonesian and other foreign languages. According to the respondents, to increase the number of Sahu Tala'i language users, especially the younger generation, they must cooperate with the village government in preserving this regional language. For example, making local language competitions, providing socialization to the community so that the Sahu Tala'i language is increasingly recognized, including the Sahu Tala'i language in local content subjects at school, developing a dictionary application, and others. Also, to be proficient in Sahu Tala'i, the respondents urged the community to practice Sahu Tala'i anywhere, especially when communicating with other family members.

In response to their needs to preserve the Sahu Tala'i language, a bilingual dictionary application, Sahu Tala'i and Indonesian, is necessary. Hence, this application would be built using the Aho-Corasick algorithm due to its speed in searching strings with many patterns.

Analysis of Potential Application Users

The potential users are those people for people who live in the Sahu tribe as well as those who are interested in learning this language for various reasons.

Definition of Requirements Specification

The following is the list of requirement specifications for the bilingual dictionary application:

1. The system should be able to save new word details into the dictionary database.
2. The system should be able to save the updated details.
3. Be able to select the origin and destination languages to translate the word you want to search.
4. Search can be done by selecting the word category or inserting keywords.
5. Display the search results by which the user can further view the word details.
6. Be able to play back the pronunciation of words spoken by both male and female voices.
7. To provide examples of how the word is used in sentences.

Analysis of the Aho-Corasick Algorithm

The following will demonstrate how the Aho-Corasick algorithm works.

***Pattern* : {Makan, Nasi, Goreng}**

Step 1: Construct the tree and pattern that has been created.

In this step, the algorithm will construct a tree from the strings to be matched. It starts with an empty root node that is known as the default non-matching state. Lindblad (2023) explained that every pattern needs to be matched to contribute to expanding the states within the machine, beginning from the root and progressing toward the pattern's end. After constructing the tree, it is traversed to establish failure pointers for each node. These pointers are directed to the longest prefix of the node that also connects to a valid node in the tree.

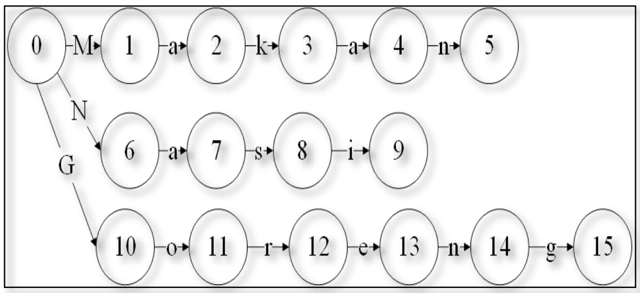
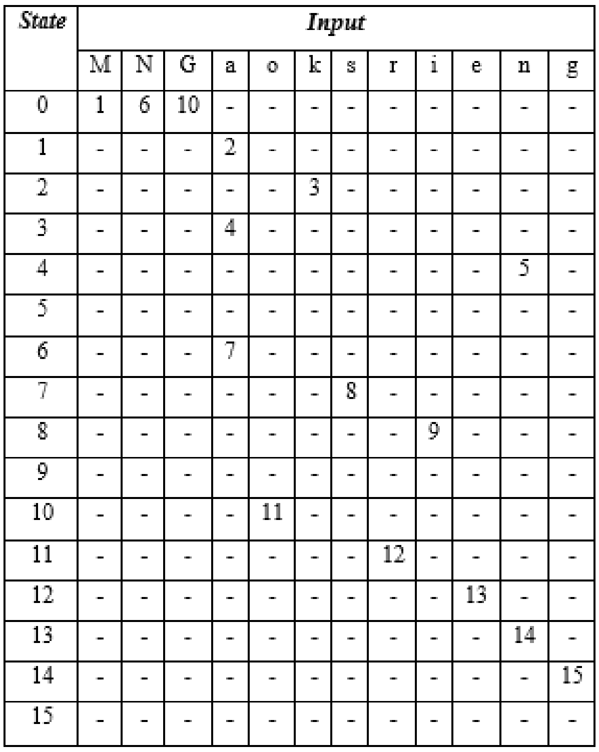


Figure 15 A Tree of Multiple Patterns

Figure 15 illustrates 16 states after determining the tree of multiple patterns "Makan, Nasi, Goreng". State means the position in the pattern according to the pattern that has been matched.

Step 2: Generate a transition table from automata.

There are unique inputs such as: "M N G a o k s r i e n g" and are used to simplify the transition table. Lindblad (2023) pointed out that each state includes a transition pointer for each character within the alphabet, alongside a record of matched patterns. Aho and Corasick illustrate the process for calculating these transition pointers as demonstrated in Figure 16.



State	*Input*											
	M	N	G	a	o	k	s	r	i	e	n	g
0	1	6	10	-	-	-	-	-	-	-	-	-
1	-	-	-	2	-	-	-	-	-	-	-	-
2	-	-	-	-	-	3	-	-	-	-	-	-
3	-	-	-	4	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	5	-
5	-	-	-	-	-	-	-	-	-	-	-	-
6	-	-	-	7	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	8	-	-	-	-	-
8	-	-	-	-	-	-	-	-	9	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	11	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	12	-	-	-	-
12	-	-	-	-	-	-	-	-	-	13	-	-
13	-	-	-	-	-	-	-	-	-	-	14	-
14	-	-	-	-	-	-	-	-	-	-	-	15
15	-	-	-	-	-	-	-	-	-	-	-	-

Figure 16 A Transition Table

Use Figure 16 to fill in the node numbers in the transition table shown in Figure 15.

a. State 0:
   1) If there is an "M" character, then move to state 1. This is the first step to find the "Makan" pattern.
   2) If there is an "N" character, then move to state 6. This is also the initial stage to find the "Nasi" pattern.
   3) If the character "G" is found, then move to state 10 which is also the initial step to find the "Goreng" pattern.

b. State 1:
   4) If there is an "a" character then move to state 2. This is the second step of the "Makan" pattern.

c. State 2:

5) If the character "k" is present, then move to state 3. This represents the third step of the "Makan" pattern.

d. State 3:

6) If the character "a" is present, then move to state 4. This represents the 4th step of the "Makan" pattern.

e. State 4:

7) If the character "n" is present, then move to state 5. This indicates the final step of the "Makan" pattern.

f. State 5:

8) There is no character in this state because it is at the end of the pattern. If it is at this stage, then it has successfully matched the "Makan" pattern.

g. State 6:

9) If there is an "a" character, then move to state 7. This is the second step to finding the "Nasi" pattern in Figure 4.

h. State 7:

10) If there is an "s" character, then move to state 8. This is the third step to find the "Nasi" pattern in Figure 4.

i. State 8:

11) If there is an "i" character, then move to state 9. This is the last step to find the "Nasi" pattern.

j. State 9:

12) In this state, there is no longer any character, because it is at the end of the pattern. If it is at this stage, then it has successfully matched the "Nasi" pattern.

k. State 10:

13) If there is an "o" character, then move to state 11. This is the second step to find the "Goreng" pattern.

l. State 11:

14) In this state can be seen in Figure 4, if there is an "r" character, then move to state 12. This is the third step to find the "Goreng" pattern.

m. State 12:

15) If there is an "e" character, then move to state 13. This is the fourth step to find the "Goreng" pattern.

n. State 13:

16) If there is an "n" character, then move to state 14. This is the fifth step to find the "Goreng" pattern.

o. State 14:

17) If there is a "g" character, then move to state 15. This is the last step to find the "Goreng" pattern.

p. State 15:

18) In this state there are no characters because it is at the end of the pattern. If you are at this stage, then you have successfully matched the "Goreng" pattern.

Step 3: Do string matching and failure function checking.

Self-checking of the failure function is done so that when no matching pattern is found, the failure function will search in each tree. This is done to ensure that no patterns are missed in the string-matching phase. The following is the pattern-matching process where there are 2 examples to find the {Eat, Rice, Fried} pattern.

A. Keywords: Makan Nasi Goreng.

Table 6 String Matching Phase "Makan Nasi Goreng"

State	Character	Transition	Description
0	M	0>1	Transition found
1	a	1>2	Transition found
2	k	2>3	Transition found
3	a	3>4	Transition found
4	n	4>5	Transition found
5		-	Transition not found
0	N	0>6	Transition found
6	a	6>7	Transition found
7	s	7>8	Transition found
8	i	8>9	Transition found
9		-	Transition not found
0	G	0>10	Transition found
10	o	10>11	Transition found
11	r	11>12	Transition found
12	e	12>13	Transition found
13	N	13>14	Transition found
14	G	14>15	Transition found
15		-	Transition not found

As can be seen again in Table 6, in state 5, where the pattern "Makan" has been successfully found, the result is "transition not found". Similarly, in state 9 the pattern "Nasi" has been successfully found, therefore the result is "transition not found". In state 15, the pattern "Goreng" has been found so the result obtained is 'transition not found'.

B. Keywords: Makan Nasa Goreng

Table 7 String Matching Phase "Makan Nasa Goreng"

State	Character	Transition	Description
0	M	0>1	Transition Found
1	a	1>2	Transition Found
2	k	2>3	Transition Found
3	a	3>4	Transition Found
4	n	4>5	Transition Found
5		-	Transition Not Found
0	N	0>6	Transition Found
6	a	6>7	Transition Found
7	s	7>8	Transition Found
8	a	8>0	Transition Not Found
0	G	0>10	Transition Found
10	o	10>11	Transition Found
11	r	11>12	Transition Found
12	e	12>13	Transition Found
13	n	13>14	Transition Found
14	g	14>15	Transition Found
15		-	Transition Not Found

Table 7 shows that there is no transition in state 5 because it has successfully matched the "Makan" pattern. Later, there is a mismatch in the characters in state 8 because the character "a" is not found in the "Nasi" pattern, and then the transition process returns to state 0, to search for the next pattern. In state 15, because the "Goreng" pattern has been found, the result is "transition not found".

Step 4: Display output function.

a. Output function keywords: Makan Nasi Goreng.

The result of the string matching displayed is: "Makan Nasi Goreng" in Figure 17.

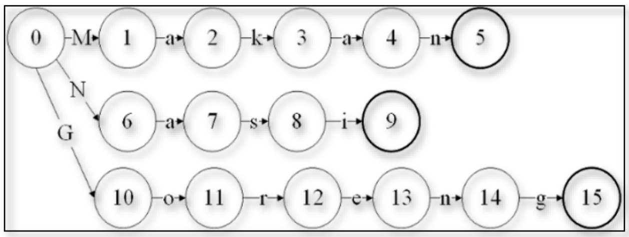


Figure 17 The output "Makan Nasi Goreng"

Table 8 The Output "Makan Nasi Goreng"

Final State	Output
Node 5	Makan
Node 9	Nasi
Node 15	Goreng

b. Output function keywords: Makan Goreng.

The result of the string matching displayed is: "Makan Goreng", as there is no "Nasa" pattern in Figure 18.

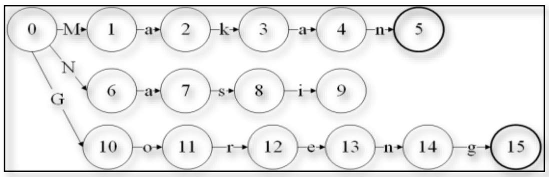


Figure 18 The output "Makan Goreng"

Table 9 Output Function

Final State	Output
Node 5	Makan
Node 15	Goreng

CONCLUSION

The implementation of the Aho-Corasick algorithm into the bilingual dictionary of Sahu Tala'i and Indonesia has been successful. This application can be used offline on a mobile phone with a minimum version of Android 4.4 KitKat. It does not require high hardware specifications like other applications as the running time of the Aho-Corasick algorithm is not affected by the number of patterns in the text. The Aho-Corasick algorithm scans each character in the text sequentially, without making any jumps or skips. Furthermore, the application is easy to use and can search up to 3 words at once. It can return the search results in no time. It also provides interesting features like giving an example of how the word is used in a sentence, the word pronunciation by male and female speakers, and word categorization. Despite this, it is highly recommended to add more rules to the algorithm to advance the search and not be limited to 3 words per search. Furthermore, it is important to populate the dictionary database with new words in both Sahu Tala'i and Indonesian to increase the accuracy of word searches.

ACKNOWLEDGMENT

We would like to thank Universitas Katolik De La Salle Manado for its continuous support that enabled us to complete this research. Also, we were grateful for the contribution of our research respondents to make this research possible.

REFERENCE

Amin, N. F., S. Garancang, and K. Abunawas. 2023. "Konsep Umum Populasi dan Sampel dalam Penelitian." *Jurnal Pilar* 14 (1): 15-31.

Bangowa, j., T. Lalu, D. Baikole, M. P. Tumbihas, and N. Loha, interview by Christ Loha. 2023. *Penggunaan Bahasa Sahu Tala'i* (Maret 15).

Budiarto, R. 2020. *Rekayasa Perangkat Lunak.* Jakarta: RBH.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2022. *Introduction to Algorithm.* 4th. Cambridge, Massachusetts: The MIT Press.

Ginting, G. L., and E Hatmi. 2018. "Implementasi Algoritma Aho-Corasick Pada Pencarian Arti Istilah Bahasa Gaul." *Majalah Ilmiah INTI* 5 (2): 138–141.

Guo, Y., Z. Qiu, H. Huang, and C. E. Siong. 2023. "Improved Keyword Recognition Based on Aho-Corasick Automaton." *2023 International Joint Conference on Neural Networks (IJCNN).* Gold Coast, Australia: IEEE. 1-7.

Indonesia, Humas Sekretariat Kabinet Republik. 2023. *Merdeka Belajar Untuk Revitalisasi Bahasa Daerah Yang Terancam* . April 23. Accessed September 1st, 2023. https://setkab.go.id/merdeka-belajar-untuk-revitalisasi-bahasa-daerah-yang-terancam/.

Jeyaseeli, A. Mary, and C. Shanthi. 2023. "Design of an Efficient Smart Phone Data Extraction Tool Using Aho-Corasick Algorithm." *Revue d'Intelligence Artificielle* 37 (2): 475-481.

Kanda, S., K. Akabe, and Y. Oda. 2023. "Engineering faster double-array Aho–Corasick automata." *Software: Practice and Experience* 53 (6): 1332-1361.

Kini, S., A. P. Patil, M. Pooja , and A. Balasubramanyam. 2022. "SQL Injection Detection and Prevention using Aho-Corasick Pattern Matching Algorithm." *2022 3rd International Conference for Emerging Technology (INCET).* Belgaum, India: IEEE. 1-6.

Lindblad, G. 2023. *Evaluating Performance of Pattern Searching Algorithms on Wildcard Patterns.* Bachelor Thesis, Department of Computing Science, UMEA University, Sweden: UMEA University, 1-21.

Loukides, G., and S. P. Pissis. 2022. "All-pairs suffix/prefix in optimal time using Aho-Corasick space." *Information Processing Letters* 178: 106275.

Pratama, P. A. E. 2021. *Komputer dan Masyarakat.* Bandung: Informatika .

Setiawati, S. 2016. "Penggunaan Kamus Besar Dalam Pembelajaran Kosakata Baku Dan Tidak Baku." *Jurnal Gramatika (Jurnal Penelitian Bahasa dan Sastra Indonesia)* 2 (1): 44-51.

Situmorang, C, A. S. Sembiring, and R Limbong. 2018. "Perancangan Aplikasi Arsip Digital Pencarian Surat Masuk dan Surat Keluar dengan Metode Aho-Corasick pada Poltekkes Kemenkes Ri Medan." *Pelita Informatika: Informasi dan Informatika* 17 (2): 133–137.

Slamet , R, and S Wahyuningsih. 2022. "Validitas Dan Reliabilitas Terhadap Instrumen Kepuasan Kerja." *Jurnal Manajemen dan Bisnis* 17 (2): 51-58.

Sommervile, I. 2016. *Software Engineering 10th ed.* USA: Pearson Education Limited.

Sulistiyowati, W, and C. C. Astuti. 2021. *Buku Ajar Statistika Dasar.* Sidoarjo: Umsida Press.

Halaman ini sengaja dikosongkan