

Rule-Based Transliteration of Ulu Kaganga Script using Character Mapping

**Ilman Zuhri Yadi¹, Yesi Novaria Kunang^{2,*}, Tia Permata Sari³, Mahmud⁴,
Nuzulur Ramadhona⁵**

^{1,2,3,4} Intelligent Systems Research Group, Faculty of Sciences Technology,
Universitas Bina Darma, Palembang, Indonesia

⁵Community of Ulu Script Enthusiasts, Palembang, Indonesia,

Email: ¹ilmanzuhriyadi@binadarma.ac.id, ²yesinovariakunang@binadarma.ac.id,

³tiapermatasari06@gmail.com, ⁴mahmud120398@gmail.com, ⁵nuzulur.romadhona@gmail.com

Abstract

Ulu Kaganga script is a historical writing tradition that developed in the southern region of Sumatra. With the widespread use of Latin script, the Ulu Kaganga script has become rare, and very few people can read and write in this script. To preserve the Ulu script, a tool is needed to assist in transliterating Latin text into the Ulu script. This research aims to preserve the Ulu script with the help of technology. In this study, a mobile and web-based application has been developed to transliterate the Ulu Kaganga script from Latin text. The technique used for this script conversion is rule-based, which is employed to break words into syllables and map those syllables into Ulu script characters. Through the rule-based technique and character mapping, adding Indonesian syllables and writing Ulu Kaganga script characters, consisting of 1139 primary characters, becomes easy. This application has been repeatedly tested to improve the mapping of Ulu script characters. The results of testing the application to transliterate 1746 words from Latin script were successful in transliterating. The tests conducted show that the approach used is very effective, with a transliteration accuracy from Latin to Ulu script of 99.98%. The testing results show that the application can transcribe text accurately and conveniently, allowing non-expert users to write in Ulu script characters.

Keywords: Character Mapping, Kaganga Ulu Script, Rule-based, Syllabification, Transliteration

1. INTRODUCTION

Indonesia is an archipelagic country with a diverse cultural heritage. One of these cultural treasures is scripts or native writings found in various regions. Among the various Nusantara scripts, Aksara Ulu is one type of ancient writing tradition that communities in southern Sumatra. This script emerged in the 12th century A.D. and experienced rapid growth during the 17th to 19th centuries A.D. [1]. The development of the Ulu script in the Uluan region or along the riversides of South Sumatra encompasses areas such as Lahat, Pagaralam, Musi Rawas, Ogan

3207



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Komerling Ulu, and also the Ilir region, including Palembang and its surroundings [1], [2], [3]. There are various variants of the Ulu script, which are derived from the Kaganga script, including the Ulu Pasemah, Ulu Serawai, Ulu Lembak, Ulu Palembang, Ulu Rejang, Ulu Lampung, and Ulu Komerling scripts [4], [5]. In the past, these scripts were commonly used for daily activities such as storytelling, writing poems, and recording customary laws [2]. The limited ability to read and write the Ulu script has led to limitations in conveying information from existing Ulu script artefacts or manuscripts.

The Ulu Kaganga script of South Sumatra is only understood by a handful of people, typically philology experts. The lack of younger generations who recognize and can read the Ulu Kaganga script is a serious concern. This is evident from the many historical documents written in Ulu Kaganga that are now difficult to comprehend and increasingly rare. The dominance of the Latin script in communication has led to the neglect of the Ulu Kaganga script, potentially resulting in the loss of this valuable cultural heritage. In this context, the tools developed in this research to assist with transliteration play a very important role. The tools will facilitate the process of converting from the Latin script to Ulu Kaganga. Thus, they provide an opportunity for younger generations to learn and understand the Ulu Kaganga script. Additionally, the accessibility of this tool enables more people to engage with their culture and history, thus contributing to the preservation efforts of cultural and linguistic heritage.

Due to its decreasing usage, the Ulu script is one of Indonesia's cultural heritages that needs to be preserved. Without efforts to introduce and preserve the Ulu script among the public, it could face the risk of extinction, as it lacks speakers from the younger generation. An approach to improve the use of the Ulu Kaganga script is to obtain the help of technology for digitization [6]. Currently, information technology is a tangible tool that facilitates various societal activities. Technological utilization for preservation has already been applied to several scripts, such as the transliteration of Bima script [7], Javanese script [8], [9], Balinese script [10], [11], Sundanese script [12], as well as scripts from other nations such as Arabizi script [13], Urdu script [14], Kurdish script [15] and Punjabi script [16].

Previous research has successfully transcribed the Komerling script, or Ulu Kaganga script, into Latin script using deep learning methods [6]. The character recognition application for the Ulu Kaganga script employs a Convolutional Neural Network (CNN) model with 96% accuracy for recognizing handwritten text and 100% accuracy for images from input photos. However, this research has not yet included the ability to translate Latin characters into Ulu script. Other studies have developed applications for recognizing Ulu script, basemah, or kaganga using simple script image mapping techniques [17], [18] - This image mapping functions as a virtual keyboard for entering required characters. The

limitation of this image mapping lies in its constrained ability to transcribe characters based solely on limited image mapping, restricting users to transcribing characters based on available images. In contrast, the Ulu script is complex because its writing system is syllable-based and differentiated by punctuation. Consequently, the variety of characters corresponding to syllables amounts to around 336 characters [6].

Hence, this research aims to develop an Android-based and web-based application for transliterating Latin text into the Ulu Kaganga script using rule-based techniques for syllable segmentation and character mapping. This technique facilitates users in directly transcribing Latin sentences or paragraphs into the Ulu script. Through the developed application, users can seamlessly transcribe Latin to Ulu without the need to search for individual script characters.

Transliteration is the process of copying and substituting alphabet letters with others without changing phonetic symbols [19]. Various previous studies have employed different methods for script transliteration. For example, research has used rule-based algorithms and the Levenshtein distance to transcribe Latin script in the Balinese language into Balinese script [10]. The rule-based approach converts Latin text into Balinese script, while the Levenshtein algorithm corrects grammar based on a database. The results show a test accuracy of 99.09% for document transcription.

Further improvement of special transliteration rules for affixing words in Balinese script was performed in [20]. The rule-based model was also used to convert Arabizi to Arabic script [13]. They identified a model from native speakers and experts' transliteration rules and utilized a discriminative model as a sequence classification task. Similarly, a rule-based approach was used to transliterate the Bima script [7], [21]. The developed transliteration application used a string replacement method to convert Bima script characters into Latin letters, employing 171 rules [21].

The Rule-Based Method is also used for the syllabification process of the Indonesian language [22]. Syllabification rules are applied to break down essential Indonesian words into syllables. Similarly, syllabification is used to segment syllables in the transliteration of Latin script into the Javanese language [23]. They claim that using rule-based syllabification reduces the complexity of creating finite-state diagrams. Syllabification algorithms are also used to spell words in the Balinese language [24]. The syllabification process, or the spelling of syllables in the Indonesian language, is crucial to convert sentences from Indonesian or Latin script to the Ulu script based on syllables.

From the studies mentioned above, the rule-based technique has been successfully implemented in many transliteration problems for languages with limited resources, yielding satisfactory results. However, it is essential to consider the complexity of rules for the diversity of the language itself. Because rule interdependence can be reasonably complicated, rule designers must cross-check whether the outcome of rule application is valid in all circumstances. This process makes developing and maintaining rule systems extremely time-consuming [13].

Therefore, in this research, we will develop a rule-based approach for the syllabification process of parsing text into syllable spellings in the Indonesian language. We will also create rule-based methods to map characters according to syllables. The generated rule-based approach will be implemented in an Android-based application for transliterating Latin script to Ulu script, aiming to facilitate user mobility.

2. METHODS

2.1. The Writing of the Ulu Script

The Ulu script of South Sumatra, also known as the Ka-ga-nga script, exhibits a variety of shapes and variations depending on its regional distribution. Some refer to this script as the Komering script, Ogan script, Rejang script, Pesemah script, and so on [5].

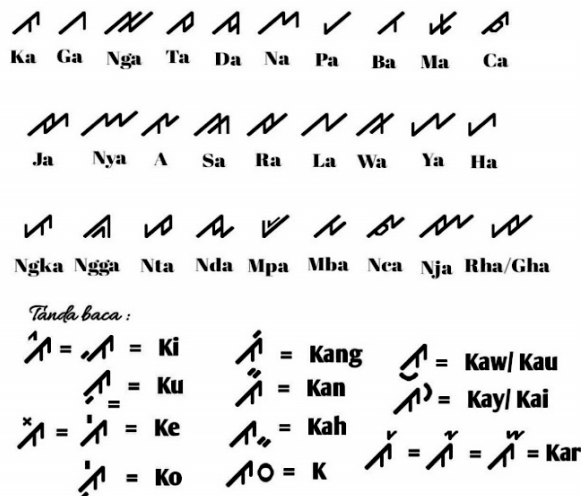


Figure 1. Letters/syllabaries of the Ulu script and diacritical markings [25]

However, it generally consists of 28 characters, including 19 main characters (ka, ga, nga, ta, da, na, pa, ba, ma, ca, ja, nya, a, sa, ra, la, wa, ya and ha) and 9 additional characters (ngka, ngga, nta, nda, mpa, mba, nca, nja, rha). The writing system used

is syllabic, with symbols representing syllables. Additionally, the Ulu script has 11 diacritical marks that change vowels, add affixes, and modify consonants [25]. The addition of diacritical marks will transform the base character into consonant syllables. For example, the character "ka" will change to "ki," "ku," "ke," "ko," "kang," "kan," "kah," "k," "kaw," "kay," and "kar." When considering the number of characters and diacritical marks, there are 1139 symbol characters for basic syllables in the Ulu script. Figure 1 shows the forms and writing of the Ulu script, along with the diacritical marks used as a reference.

2.2. Method of Transliterating Latin Script to Ulu Script

Rule-based mappings from Latin script to Ulu script involve establishing specific correspondences for sounds, letters, and phonetic characteristics unique to Indonesian. For instance, the Latin letter 'a' is mapped to the Ulu character, because 'a' generally represents an open front vowel sound, which is similarly represented in Ulu. The consonant 'k' corresponds voiceless velar plosive, Ulu contains a direct equivalent conveying the same phonetic value. In cases of diphthongs, the Latin 'ai' is represented in Ulu, capturing the blended vowel sound that 'ai' denotes. Similarly, the Latin 'ng' is mapped to Ulu's, which corresponds to a nasal consonant sound that exists in both scripts. For palatalization, the Latin 'c' becomes 'k' in Ulu when followed by 'i' or 'e', as 'c' often represents a palatal sound (like 'ch') in such contexts.

The rationale for these mappings primarily hinges on phonetic correspondence, ensuring that each letter or combination of letters is matched with its closest equivalent in Ulu script based on sound. Additionally, cultural considerations are factored in, particularly regarding phonemes that may exist in bahasa but not in another, emphasizing similarities rather than exact matches when needed. This approach also aids in learning by providing intuitive mappings based on familiar sounds and characters, facilitating a smoother transition for learners shifting between the two scripts. The process flow of transliterating Latin script into Ulu script using this rule-based technique consists of three stages: syllabification, character mapping, and the sequential search process to find the corresponding Ulu script character images (Figure 2).

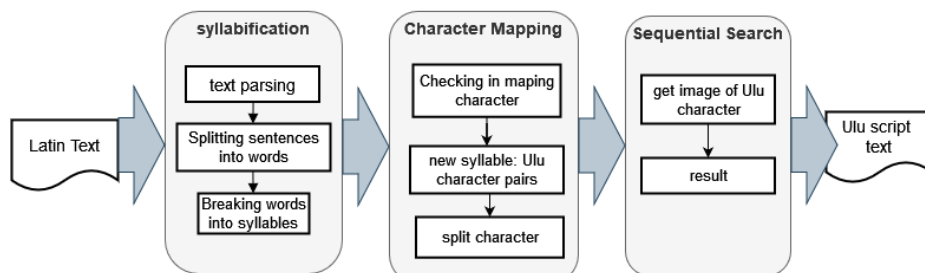


Figure 2. Flow of the transliteration process from Latin script to Ulu

The explanation of each step is as follows.

- 1) The initial process begins with the user entering a sentence or paragraph written in Latin script, the process as shown in Algorithm 1.

Algorithm 1. Syllabification algorithm

```
Input: word
Output: syllables
Call the syllabify function with the provided word.
Inside the syllabify function:
1. Call the replacer function with the given word to get a modified
   version of the word.
2. Call the preprocess function with the modified word to obtain a list
   of syllables.
3. Call the process function with the list of syllables to further
   refine them.
4. Call the unreplace function with the processed syllables to revert
   any replacements made.
5. Print the unreplaced processed syllables.
6. Return the unreplaced processed syllables.
7. End
```

- 2) The initial stage involves syllabification. The Latin text is entered into the system and broken down into words. Each word undergoes syllable segmentation. The syllabification process can be seen in Algorithm 1, which modifies algorithm [23]. The result of this syllabification process transforms the sentence into a sequence of syllabic spellings according to the rules of the Indonesian language. An example of a sentence resulting from the syllabification process will transform the sentence "aku cinta indonesia" into "a-ku-cin-ta-in-do-ne-si-a." The results of the syllable segmentation are stored in an array, the process as shown in Algorithm 2.

Algorithm 2. Character mapping algorithm

```
Input: text
Output: response
1. Create a new `Process` to run a Python script with the `$text` as an
   argument.
2. Run the process.
3. Check if the process is not successful:
   If not successful, throw an exception and print the error
   output.
4. Get the process output and normalize it into the `$splitAksara`
   member.
5. Normalize the `$splitAksara` by adding characters as needed.
6. Replace characters 'f', 'v', 'q', 'x', 'z' with their respective
   updates.
7. Retrieve Aksara information from the database based on the
   normalized characters.
8. Normalize the database result and prepare the response.
9. Set the `$response` member with the normalized result.
10. End.
```

- 3) After syllable segmentation, the second stage involves character mapping based on the generated syllables. The algorithm for this stage can be seen in Algorithm 2. After syllable segmentation is performed using the `splitAksara` function, it continues with the `normalizeCharacter` function. This function searches for syllables in the normalization character table and stores the character breakdown results from the syllables. Some Latin letters that do not exist in the Ulu script are mapped or replaced with characters, namely ['f' => 'p', 'v' => 'p', 'q' => 'k', 'x' => 'k', 'z' => 'j']. Then, the syllables and character breakdowns are mapped to Ulu characters by the `get_aksara` function, which searches for syllables in the aksara table, the process as shown in Algorithm 3.

Algorithm 3. Sequential searching for image character algorithm

Input: `normalize_character`

Output: `this`

1. Initialize an empty array called `$result`.
 2. Call the `'getAksara'` method of a new instance of the `'Aksara'` class, passing the result of `'$normalize_character->toArray()'` as an argument. Store the result in the variable `$result`.
 3. Map over each element in the `$normalize_character` collection using a closure function that takes an `$aksara` as a parameter and uses the `$result` obtained in step 2:
 - a. For each `$aksara`, return a filtered result from `$result` where the `'keterangan'` attribute matches the current `$aksara`.
 - b. Flatten the resulting nested array structure by one level using `'flatten(1)'`.
 4. Store the flattened result in the variable `$resultAksara`.
 5. Map over each item in `$resultAksara` using a closure function:
 - a. Get the asset URL for `'data_aksara'` and store it in `$assetUrl`.
 - b. Check if the `'file'` attribute of the current `$item` does not start with `$a setUrl` using `'Str::startsWith'`.
 - c. If the `'file'` attribute doesn't start with `$assetUrl`, update the `'file'` attribute by appending `$assetUrl` and a slash `('/')` to the beginning of the `'file'` attribute.
 - d. Remove the `'created_at'` and `'updated_at'` fields from the `$item` using `'Arr::forget'`.
 - e. Return the modified `$item`.
 6. Set the property `$this->response` to the value of `$resultAksara`.
 7. Return the current object instance (`$this`) to allow method chaining.
 8. End.
-

- 4) The final stage involves finding images corresponding to the sequence of syllables and character splits in the aksara table. This table contains 1139 Ulu script character images, and the algorithm for this can be seen in Algorithm 3. The search process uses sequential search techniques based on queries in the table to find character names based on the `'keterangan'` (description) field. It then retrieves the image file name from the database on the `'file'` field. Subsequently, from the file name, the program displays the Ulu script image stored in the `'data_aksara'` folder.

- 5) The final step will present the results of the transliteration process in the form of Ulu script characters.

2.3. System Development Architecture

The development architecture of the Latin script to Ulu script transliteration application can be seen in Figure 2. A cloud server for the application is deployed using Docker, which contains several tools. Laravel is used as the framework for developing PHP-based service applications. The Laravel framework simplifies the development of PHP applications that connect to a MySQL database and applications developed in other programming languages, such as Python. Laravel is also used to generate PHP Laravel REST APIs to connect with the Android client application. The PHP REST API is used for applications that map syllables to Ulu script symbols. Additionally, Python scripts are developed for syllabification and sentence parsing.

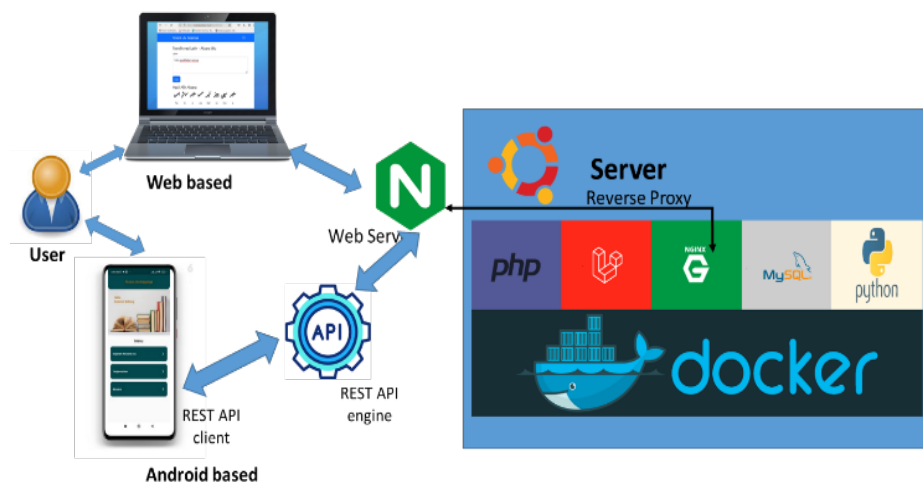


Figure 3. Development architecture for transliteration system from Latin script to Ulu script

MySQL is the database used to store syllable mappings with Ulu script file names. Furthermore, the Nginx server serves as a Web server and functions as a reverse proxy. Nginx handles requests to the web server and maps the service requests based on ports. The developed application consists of two platforms: a web-based application and an Android-based application. The Android-based client application requests services through the Nginx reverse proxy server. Meanwhile, administrative processes such as adding character maps, checking syllabification processes, and adding Ulu script characters are accessed through the Web.

3. RESULTS AND DISCUSSION

3.1. Ulu Kaganga Transliteration Application Testing

The application has been developed on two platforms, web-based and Android-based. The android based transliteration application for transliterating Latin script to Ulu script has been tested on Android platforms version 9 and above. The application itself can be downloaded from the following link: <https://bit.ly/nulisksaraUlu>. This transliteration capability will be integrated into the intelligent Ulu script recognition application, which previously could only detect Ulu script [26]. This new feature for Ulu script transliteration will enable the application to convert Latin characters to Ulu script. The web-based platform is used for application testing and requires admin involvement to map syllabic changes and add images of Ulu Kaganga character. To access the web-based application, visit <https://nulisaksaraulu.my.id/>. Meanwhile, the Android-based application is designed for user convenience and mobility.

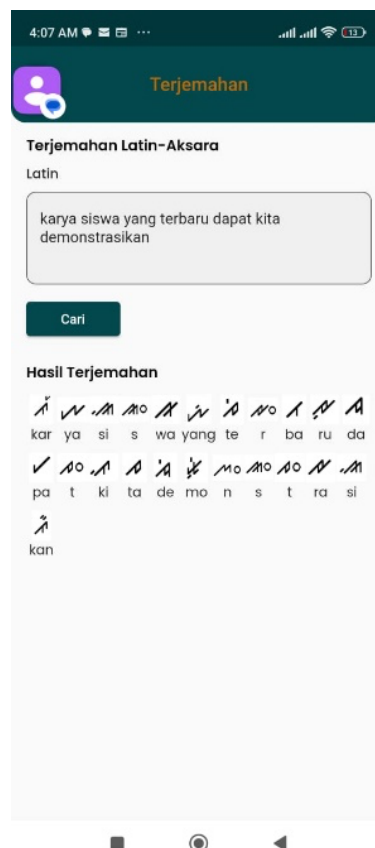


Figure 4. The Ulu transliteration process in android-based application

The Android-based menu interface for transliterating Latin script to Ulu script presents three main menus: Ulu script history, translation, and the characters in the Ulu script. The Ulu scripts history menu contains historical information on the Ulu script. Additionally, there is a Ulu script menu to help users understand the primary syllabic characters of the Ulu script. This menu retrieves data from the script table, displaying all forms of script present based on the file names within the table. Administrative features are required to make changes to images or symbols.

The translation menu in android-based application is used to translate Latin script into Ulu script. Users simply enter a sentence or word to be transliterated, then click the search button, and the translation results will be displayed, as shown in Figure 4. The client application communicates with the REST API engine to request services from TranslateService.php. When executed, the syllabification process breaks the text into syllables, followed by character mapping, and finally, the translation results are displayed. Not much different for the web-based Ulu script transliteration application also utilizes a REST API client running in the browser that communicates with the TranslateService REST API engine. The interface of the client application itself can be seen in Figure 5.

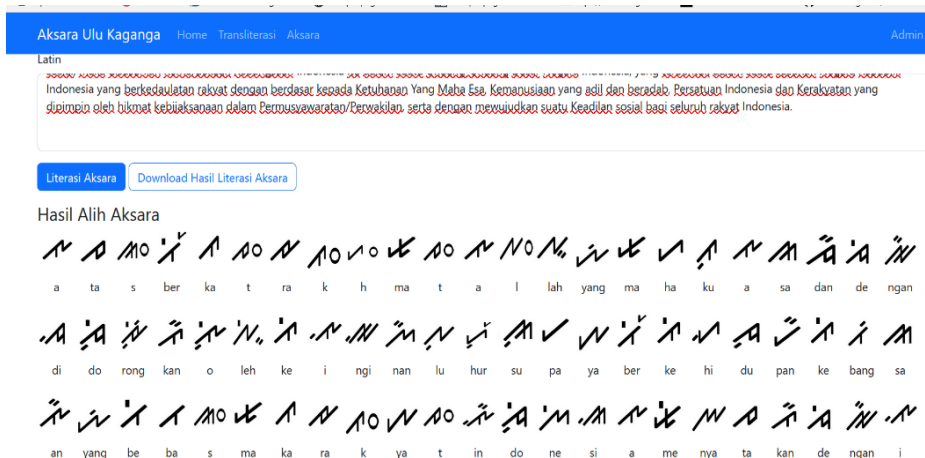


Figure 5. The Ulu transliteration process in web-based application

3.2. Application Administration Features

The administration process primarily focuses on adding data for Ulu script symbol characters and mapping syllables with symbol characters from the Ulu script. Users can log in as an admin to access the addition or editing of Ulu script character images. Additionally, there is a normalization menu for the syllable mapping process to derive basic syllables from the Ulu script. This menu enhances syllable

mapping by adding pairs of 2 or 3 characters from Ulu script syllables, as shown in Figure 6. This research has gradually mapped 1139 basic Ulu script syllables and 1631 mapping syllables.

The rule-based syllabification process will break down sentences or text into syllables in the language. The evaluation process for syllable segmentation, according to the rules of the Indonesian language, can be observed in Figure 7. As depicted in the application, it effectively dissects syllables. This step is essential to assess whether the application is accurately presenting transliteration results.

3.3. Transliteration Results Testing

To evaluate the transliteration results, tests were conducted by inputting test text. The evaluation aimed to assess how accurately the developed application performs transliteration. The testing was carried out progressively. In the first phase, a trial was conducted by transliterating several sentences obtained from the internet and pasted into the application. The tested data consisted of text and sentences. During the initial testing, there were 95-character mappings available in the normalization table. In this first round of testing, 20 trials were conducted, inputting a total of 775 words. There were 7 failed attempts to transliterate, resulting in 15 syllables that could not be successfully converted into Ulu script characters. These 15 split syllables with mappings not present in the normalization table were added to the database and retested. After this, all 775 words were successfully transliterated accurately.

Aksara Ulu Dashboard				Home	Data aksara	Normalisasi aksara	Literasi Aksara	Demo Split Aksara	Logout
1	tru	t ru	Edit	Hapus					
2	khi	k hi	Edit	Hapus					
3	sta	s ta	Edit	Hapus					
4	tra	t ra	Edit	Hapus					
5	gra	g ra	Edit	Hapus					
6	pre	p re	Edit	Hapus					
7	rat	ra t	Edit	Hapus					
8	ak	a k	Edit	Hapus					
9	nal	na l	Edit	Hapus					
10	jum	ju m	Edit	Hapus					

Menampilkan 1 - 9 dari 1631 hasil pencarian

1 2 3 4 5 6 7 8 9 10 ... 163 164

Figure 6. Mapping of Ulu Script character pairs

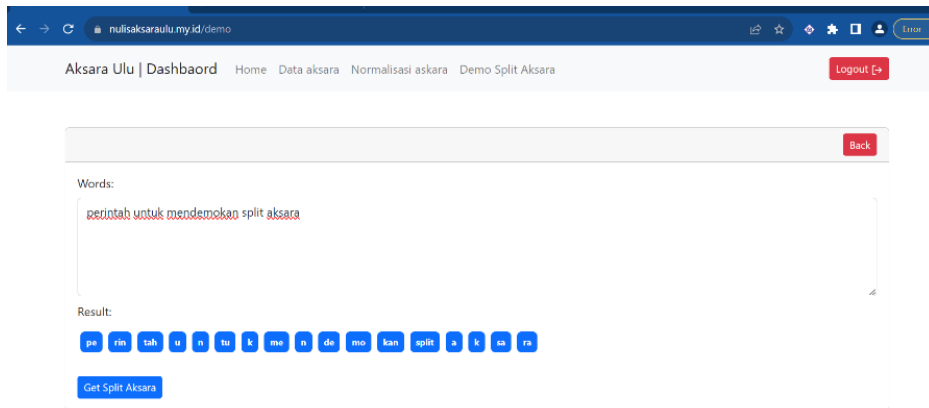


Figure 7. Split Script menu to verify the syllable segmentation process before mapping into characters

Then, additional character mappings were added for the normalization of syllables by completing the list of Indonesian syllables with a total of 2148 syllables, as referenced in the source [23]. Some of these syllables were already included in the basic syllables of the Ulu script, so the addition of normalized syllables resulted in 1631-character mappings. Afterwards, more comprehensive testing was carried out by inputting texts with varying sentence lengths for transliteration. In the second phase of testing, sentences were derived from the 1945 Constitution. Thirty trials were conducted by inputting sentences from excerpts of the preamble and the articles of the 1945 Constitution. The testing results are presented in Table 1. Testing from 1 to 25 utilized the main body from the preamble to its articles. Additionally, in tests 26 to 30, several revised articles from the fourth amendment of the 1945 Constitution were used as the input text.



Figure 8. Test 2 resulted in one syllable not being detected

The results of the transliteration tests show that out of 30 tests, with a total of 1746 input words, the application was able to transliterate all words effectively. Only one syllable was not detected out of a total of 4839 syllables resulting from the syllabification process. The application successfully segmented sentences into syllables and transliterated them into Ulu script characters with an accuracy of

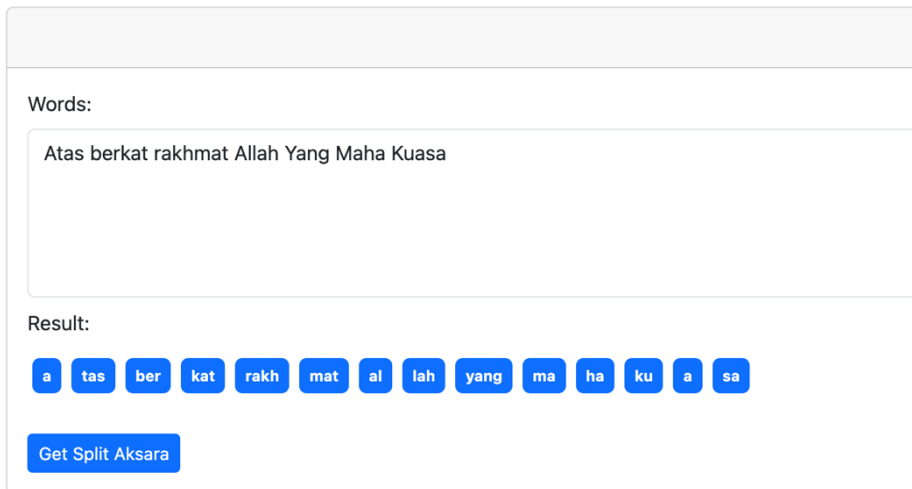
99.98%. This accuracy value is calculated based on the total number of syllables that could be transliterated compared to the overall total number of syllables resulting from the syllabification process. The result indicates that the application can perform as intended for web-based and android-based applications. During its development, the application received guidance from experts in Ulu script philology from the Ulu script enthusiast's community. Experts in Ulu script philology also conducted tests and evaluations to avoid script writing errors.

Table 1. The results of transliteration testing

Testing Number	Total Words	Total Syllables	Total Ulu Script	Undetected syllables	Accuracy
1	75	198	210	0	100%
2	121	368	399	1	99.73%
3	29	74	82	0	100%
4	69	198	230	0	100%
5	92	254	276	0	100%
6	133	376	426	0	100%
7	80	216	233	0	100%
8	30	77	82	0	100%
9	26	78	77	0	100%
10	39	116	125	0	100%
11	59	161	174	0	100%
12	81	215	237	0	100%
13	84	236	252	0	100%
14	43	118	123	0	100%
15	35	91	96	0	100%
16	53	141	152	0	100%
17	31	81	89	0	100%
18	26	66	75	0	100%
19	32	96	101	0	100%
20	63	164	175	0	100%
21	21	56	53	0	100%
22	39	106	112	0	100%
23	38	109	112	0	100%
24	43	126	142	0	100%
25	41	109	128	0	100%
26	97	281	300	0	100%
27	35	104	107	0	100%
28	60	154	174	0	100%
29	104	300	518	0	100%
30	67	170	194	0	100%
Total	1746	4839	5454	1	99.98%

The results of Test 2 for the syllabification process indicated that there was one syllable that was not detected (as shown in Figure 8).

In Test 2, the syllable "rakh" was unable to be transliterated. From the syllabification process illustrated in Figure 9, it can be observed that the syllabification algorithm was functioning effectively. Therefore, the error was due to the application not finding the appropriate character mapping for that syllable. In Figure 10, an additional syllable mapping is presented in the "normalization" table, where the syllable "rakh" is mapped as "ra-k-h."



Words:

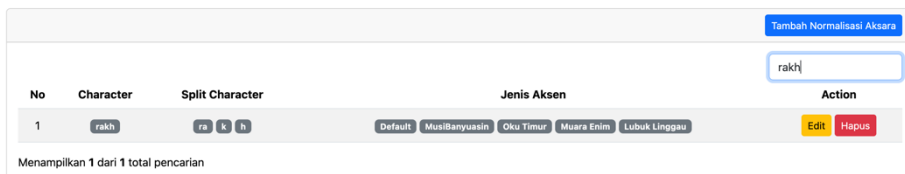
Atas berkat rakhmat Allah Yang Maha Kuasa

Result:

a tas ber kat rakh mat al lah yang ma ha ku a sa

Get Split Aksara

Figure 9. The results of syllabification of the sentence from Test 2



No	Character	Split Character	Jenis Aksen	Action
1	rakh	ra k h	Default Musibanyasin Oku Timur Muara Enim Lubuk Linggau	Edit Hapus

Menampilkan 1 dari 1 total pencarian

Figure 10. The addition of the syllable "rakh."



Hasil Formated Transliterasi

a ta s ber ka t ra k h ma t a l lah yang ma ha ku a sa

Tutup Unduh Transliterasi

Figure 11. Improvements in the transliteration results from Test 2

After the addition, the results of the transliteration show the correct outcomes as desired (Figure 11). From this testing, one of the weaknesses of this application is the necessity for continuous testing to identify deficiencies in character mapping,

particularly for syllable normalization. Currently, a total of 1,641 syllable mappings have been incorporated, in addition to 1,139 syllables derived from Ulu script characters.

User feedback and expert evaluations play a crucial role in the refinement of transliteration tools, leading to improvements in accuracy, usability, and overall effectiveness. Users often identify errors that developers may overlook, and feedback regarding incorrect transliterations can help pinpoint specific patterns or exceptions that need adjustment. Additionally, user feedback highlights usability challenges, such as complicated interfaces or unclear instructions, while expert evaluations identify best practices in user experience design. Moreover, contextual adaptation is vital, Ulu script have unique rules for transliteration. User insights often reveal gaps in addressing cultural or linguistic nuances, allowing for more contextually appropriate outcomes. Suggested features from experts can enhance the tool's functionality, including options for customizable transliteration settings, and visual aids to assist users in understanding the mapping process. Engaging with users and experts fosters a sense of community around the tool, encouraging ongoing conversations about best practices in transliteration and prompting users to share their experiences and suggestions for improvement.

However, there are some shortcomings and limitations behind the benefits and advantages of the Latin to Ulu script application. One weakness of this application is the lack of support for numbers and punctuation marks. When parsing text, punctuation marks and numbers are removed first. This limitation is due to the constraints of writing in the Ulu script, which does not include numbers and punctuation marks. Using numbers in various Ulu script variants does not yet have symbols for writing numbers. From the testing conducted using articles containing Roman numerals, they will be transliterated as characters without meaning. This can be understood because the simplicity of writing rules in ancient times is not as complex as the rules of writing in the Latin script. The Ulu script, derived from the Ka-ga-nga script, has many variants that this Latin may not yet represent to Ulu script transliteration application. Therefore, it would be interesting to develop Ulu script transliteration with various Ulu script or Kaganga script variations used in the South Sumatra region in the future.

3.4. Discussion

The results demonstrate that the developed Ulu Kaganga transliteration application, available on both web-based and Android platforms, functions effectively in converting Latin script to Ulu script. The transliteration accuracy of 99.98% highlights the robustness of the implemented rule-based syllabification and character mapping process. The successful segmentation and transliteration of 4839 syllables out of 4840 input syllables confirm the system's reliability. However,

the detection failure of the syllable "rakh" in Test 2 underscores the importance of continuously expanding the normalization table to accommodate additional syllabic mappings. The subsequent correction and successful transliteration after updating the mapping table demonstrate the adaptability of the system to incremental refinements.

The transliteration process relies on a REST API-based architecture, ensuring seamless communication between the client application and the TranslateService engine. Both the Android and web-based versions follow a similar approach, utilizing syllabification, character mapping, and subsequent transliteration. The administrative features further enhance the system by allowing the addition and modification of Ulu script characters, ensuring the application remains adaptable to new linguistic insights. Currently, the system includes 1,139 basic Ulu script syllables and 1,641 mapped syllables, demonstrating significant progress in building a comprehensive transliteration framework.

User feedback and expert evaluations are crucial in refining the application, addressing both transliteration accuracy and usability. Users can identify errors that may not be immediately apparent to developers, while expert evaluations provide deeper linguistic insights to improve the transliteration rules. This iterative feedback process has proven effective in enhancing accuracy and contextual adaptation, particularly given the unique transliteration rules of the Ulu script. Furthermore, expert recommendations can contribute to future improvements, such as customizable transliteration settings and visual aids to support users in understanding the mapping process.

Despite its advantages, the application has limitations. One notable drawback is the lack of support for numbers and punctuation marks. Currently, numbers and punctuation are removed before transliteration due to the historical constraints of the Ulu script, which does not inherently include symbols for these elements. Consequently, when transliterating text containing Roman numerals, the output may result in characters without meaning. This limitation reflects the simplicity of ancient writing conventions compared to modern Latin script rules. Additionally, the Ulu script comprises multiple regional variants, which are not yet fully incorporated into the current application. Future developments should explore transliteration models that accommodate variations of the Ulu script or Kaganga script used in the South Sumatra region. Expanding the system to support these variations would enhance its applicability and cultural relevance. Overall, the Ulu Kaganga transliteration application represents a significant step toward preserving and digitizing the Ulu script. However, continued development, guided by user feedback and expert insights, will be essential for further improving transliteration accuracy, expanding character mappings, and incorporating additional script variants.

4. CONCLUSION

In this research, we propose an application to transliterate Latin text into Ulu Kaganga script using a rule-based technique and character mapping. The proposed rule-based approach breaks down words into syllables and maps syllables to Ulu script characters. The results demonstrate that the application can easily transliterate entered text or sentences with this technique. Some challenges caused by certain syllables that cannot be displayed can be addressed by adding character mapping rules to the table. Thus, the more rules for mapping new syllables that are added, the application's ability to transliterate will improve. Furthermore, using character mapping rules simplifies the application workflow, as it only needs to search for character mappings sequentially in the table, making the mapping process more straightforward.

Potential future work may include adding more character mappings to enhance the transliteration accuracy for a wider range of words and syllables. Additionally, supporting numbers and punctuation would provide a more comprehensive transliteration system, while exploring different variants of the Ulu Kaganga script would ensure compatibility and representation of diverse linguistic features. Additionally, we will gradually enhance the transliteration application for the Ulu Kaganga script with various Ulu script variants, such as Ulu Lubuk Linggau, Ulu Pasemah, Ulu Semendo, Ulu Ogan, and others, to preserve the ancient writing culture of the South Sumatra region.

For the utilization of the developed tool, potential collaborations with cultural heritage organizations and educational institutions offer opportunities for broader dissemination of the transliteration tool. Partnering with cultural heritage organizations can promote the tool as a resource for preserving and documenting linguistic diversity, facilitating its integration into various cultural initiatives and projects. Additionally, collaboration with educational institutions can lead to the inclusion of the tool in local content curricula, workshops, and training sessions, thus reaching a wider audience of students and educators. These partnerships not only enhance the tool's visibility but also contribute to its practical application in real-world contexts, fostering greater engagement and understanding of the Ulu script and the cultures it represents.

REFERENCES

- [1] E. Rochmiatun, "Naskah Gelumpai di Ulu Palembang: antara Ajaran Islam dan Ajaran Hindu-Buddha," *Manuskripta*, vol. 9, no. 1, pp. 45–67, 2019.

- [2] M. A. Ridhollah, N. U. Kalsum, and S. Khudin, "Naskah Ulu: Obat-Obatan Tradisional Dalam Naskah Kaghas Nomor. Inv 07. 47 Koleksi Museum Negeri Sumatra Selatan (Kajian Filologi)," no. 3, 2021.
- [3] W. R. Andhifani and N. Rahmadhona, "Naskah Ulu Puyang Bang Mangu': Sebuah Batas Wilayah," *J. Penelit. Arkeol. Papua Dan Papua Barat*, vol. 13, no. 1, pp. 71–86, Jun. 2021, doi: 10.24832/papua.v13i1.298.
- [4] Y. Asmara, "Lubuklinggau's Ulu Alphabet and Its Preservation," *Istor. J. Pendidik. Dan Ilmu Sej.*, vol. 15, no. 1, Mar. 2019, doi: 10.21831/istoria.v15i1.24156.
- [5] N. Anida, N. U. Kalsum, and O. Otoman, "Suntingan dan Analisis Isi Teks Aksara Ulu dalam Koleksi Peti 91/E6," *Tanjak Sej. Dan Perad. Islam*, vol. 1, no. 2, pp. 42–53, 2021, doi: 10.19109/tanjak.v1i2.9374.
- [6] Y. N. Kunang, I. Z. Yadi, Mahmud, and M. Husin, "A New Deep Learning-Based Mobile Application for Komering Character Recognition," in *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Dec. 2022, pp. 294–299. doi: 10.1109/ISRITI56927.2022.10053072.
- [7] A. Aranta *et al.*, "Learning media for the transliteration of Latin letters into Bima script based on android applications," *J. Educ. Learn. EduLearn*, vol. 15, no. 2, pp. 275–282, May 2021, doi: 10.11591/edulearn.v15i2.19013.
- [8] V. Atina, S. Palgunadi, and W. Widiarto, "Program Transliterasi Antara Aksara Latin dan Aksara Jawa dengan Metode FSA," *J. Teknol. Inf. ITSmart*, vol. 1, no. 2, p. 60, Mar. 2016, doi: 10.20961/its.v1i2.592.
- [9] B. E. Praheto and F. B. B. Utomo, "Transliteration Method In Learning Reading Of The Javanese Script," in *International Conference on Education 2019*, PGSD Universitas Sarjanawiyata Tamansiswa, 2019, pp. 32–36.
- [10] M. Sudarma, I. N. S. Kumara, and I. Udayana, "Transliteration Balinese Latin Text Becomes Aksara Bali Using Rule Base And Levenshtein Distance Approach," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 2, no. 3, pp. 401–408, 2016.
- [11] G. Indrawan, I. G. Aris Gunadi, M. Santo Gitakarma, and I. K. Paramarta, "Latin to Balinese Script Transliteration: Lessons Learned from the Computer-based Implementation," in *2021 The 4th International Conference on Software Engineering and Information Management*, Yokohama Japan: ACM, Jan. 2021, pp. 171–175. doi: 10.1145/3451471.3451499.
- [12] Y. A. Gerhana, M. F. Padilah, and A. R. Atmadja, "Comparison of Template Matching Algorithm and Feature Extraction Algorithm in Sundanese Script Transliteration Application using Optical Character Recognition," *J. Online Inform.*, vol. 5, no. 1.
- [13] A. Masmoudi, M. E. Khmekhem, M. Khrouf, and L. H. Belguith, "Transliteration of Arabizi into Arabic Script for Tunisian Dialect," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 19, no. 2, pp. 1–21, Mar. 2020, doi: 10.1145/3364319.

- [14] H. M. Shakeel, R. Khan, and M. Waheed, "Context based Roman-Urdu to Urdu Script Transliteration System," *ArXiv Prepr. ArXiv210914197*, 2021.
- [15] S. Ahmadi, "A rule-based Kurdish text transliteration system," *ACM Trans. Asian Low-Resour. Lang. Inf. Process. TALLIP*, vol. 18, no. 2, pp. 1–8, 2019.
- [16] K. D. Goyal, M. R. Abbas, V. Goyal, and Y. Saleem, "Forward-backward Transliteration of Punjabi Gurmukhi Script Using N-gram Language Model," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 22, no. 2, pp. 1–24, Mar. 2023, doi: 10.1145/3542924.
- [17] A. Sonita and A. Susanto, "Implementasi Augmented Reality (AR) Sebagai Media Pengenalan Aksara Ka Ga Nga Rejang Lebong Berbasis Android," *J. Komput. Inf. Dan Teknol. JKOMITEK*, vol. 2, no. 2, Dec. 2022, doi: 10.53697/jkomitek.v2i2.867.
- [18] D. Setiawati, "Media Pembelajaran Pengenalan Aksara Besemah Pada Anak Sd Di Kota Pagaralam Berbasis Android (Studi Kasus: Sd N 55)," PhD Thesis, Universitas Komputer Indonesia, 2019.
- [19] J. K. L. Dimpudus, A. M. Sambul, and A. S. M. Lumenta, "Transliteration Block Notation Application Into Number Notation Using The MusicXML Format".
- [20] G. Indrawan and others, "A Method for the Affixed Word Transliteration to the Balinese Script on the Learning Web Application," *Turk. J. Comput. Math. Educ. TURCOMAT*, vol. 12, no. 6, pp. 2849–2857, 2021.
- [21] A. Aranta, F. Bimantoro, and I. P. T. Putrawan, "Penerapan Algoritma Rule Base dengan Pendekatan Hexadesimal pada Transliterasi Aksara Bima Menjadi Huruf Latin," *J. Teknol. Inf. Komput. Dan Apl. JTika*, vol. 2, no. 1, pp. 130–141, Mar. 2020, doi: 10.29303/jtika.v2i1.96.
- [22] A. Rahman, D. T. Murdiansyah, and K. M. Lhaksmana, "Silabifikasi Menggunakan Metode Rule-based Dalam Bahasa Indonesia," *EProceedings Eng.*, vol. 8, no. 5, 2021.
- [23] A. W. Mahastama, "Model Berbasis Aturan untuk Transliterasi Bahasa Jawa dengan Aksara Latin ke Aksara Jawa," *J. Buana Inform.*, vol. 13, no. 02, pp. 146–154, Oct. 2022, doi: 10.24002/jbi.v13i02.6526.
- [24] G. B. P. Putra and N. A. Sanjaya Er, "Syllabification of Balinese Words Using the Syllabification Algorithm," *JELIKU J. Elektron. Ilmu Komput. Udayana*, vol. 8, no. 2, p. 125, Jan. 2020, doi: 10.24843/JLK.2019.v08.i02.p03.
- [25] N. Ramadhona, *Buku Pedoman Aksara Ulu Sumatera Selatan*. KBM indonesia, 2022.
- [26] Y. N. Kunang, I. Z. Yadi, M. Mahmud, and M. Husin, "Aplikasi Perangkat Lunak Pendeteksi Aksara Komerling/Aksara Ulu Berbasis Android," *EC00202273702*, Oct. 2022