Improving the Time Efficiency of a Script Identification Algorithm Using a Unicode-Based Regular Expression Matching Strategy

Mamtimin Qasim * and Wushour Silamu

Posted Date: 11 December 2025

doi: 10.20944/preprints202512.1082.v1

Keywords: script; script identification; Unicode; language identification

 doi:10.20944/preprints202512.1082.v1

Article

Improving the Time Efficiency of a Script Identification Algorithm Using a Unicode-Based Regular Expression Matching Strategy

Mamtimin Qasim [1,*] and Wushour Silamu [2,3]

[1] School of Engineering, Nanfang College Guangzhou, Guangzhou 510970, China
[2] School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[3] Key Multi-lingual Laboratory of Xinjiang, Urumqi 830046, China
* Correspondence: mamtimin116@163.com

Featured Application

The proposed technology can improve the efficiency of acquiring, analyzing, and processing multilingual web text resources based on their script and language.

Abstract

Script identification is the first step in most multilingual text processing systems. To improve the time efficiency of language identification algorithms, it is first determined whether there is content written in a certain script in the text; if so, the content written in that script is then obtained. Then, it is determined whether the total length of the texts corresponding to the identified scripts is equal to the original text length; if so, the script identification process ends. Finally, considering the frequencies of various scripts on the Internet, those that appear more frequently are prioritized during script identification. Based on these three approaches, an improved script identification algorithm was designed. A comparison experiment was conducted using sentence-level text corpora in 261 languages written in 24 scripts. The training and testing times of the newly proposed method were reduced by 8.61- and 8.56-fold, respectively, while the F1 score for script identification was slightly higher than those reported in our earlier studies. The method proposed in this study effectively improves the time efficiency of script identification algorithms.

Keywords: script; script identification; Unicode; language identification

1. Introduction

Script identification (SI) is a technology in which computers are used to determine the script in which text content is written. It is the first step in most text processing systems, especially in those designed for language identification (LI)—that is, the process of determining which language a piece of text is written in and classifying it as a pre-set language [1,2]. LI is the first step in many text-processing tasks, including information retrieval, search engines, machine translation, and speech synthesis [3–6]. SI is easier to implement than LI and has a higher accuracy, but reducing the number of recognized languages in LI systems can improve their efficiency [3,4]. Therefore, some researchers have proposed techniques in which the script is first recognized and then the language within the same script is identified, effectively improving the efficiency of LI [7–9]. With the explosive growth in online information, the timely collection of multilingual text content and its language classification has become a primary problem. Therefore, it is necessary to improve the time efficiency of SI systems.

In the Unicode encoding scheme, a separate area is allocated for each script's characters and symbols (relevant information can be found on the official Unicode website [10]). Generally, SI algorithms are designed using the Unicode encoding scheme's features [11,12]. In preliminary research, we analyzed the common and specific parts of all 169 scripts on the Unicode website,

 doi:10.20944/preprints202512.1082.v1

constructed regular expressions for each script and based on this, designed an SI algorithm which is capable of recognizing these scripts while ensuring the integrity of the information in the SI results [13]. Although the focus of this preliminary research was improving the accuracy of SI, time efficiency enhancements were not considered.

In this study, the regular expressions constructed after analyzing the Unicode website scripts were applied and the following two optimization strategies were combined to design an SI algorithm with improved time efficiency.

1) According to statistics reported by Statista [14], the number of web pages in different languages varies, with some languages having a higher proportion than others. Using these statistics, we can deduce which scripts are used on more webpages. When designing a script recognition algorithm, the scripts used in most web pages should be considered first, as they are more likely to occur. Once the target script is identified, there is no need to continue analyzing the remaining scripts.
2) Sometimes, the text (or even individual sentences) contains content in different scripts. In these cases, it is necessary to identify the different scripts within the text. The length of the text is fixed, and the sum of the lengths of text in different scripts is equal to the total length of the text. Therefore, when designing a script recognition algorithm, it is necessary to analyze the lengths of different script contents being recognized: if the sum of the lengths of contents in different scripts is equal to the total length of the text, there is no need to analyze the remaining scripts.

2. Data Description

2.1. Unicode Character Database

At present, the Unicode website's character database provides encoding blocks for 169 scripts [15], as illustrated in Figure 1. In addition to the encodings for each script, it also contains symbols and characters shared by multiple scripts, categorized as "Common," as shown in Figure 2. Analysis of the web pages that provide the data tables reveals that, except for lines starting with "#", the remaining lines all contain information related to Unicode characters, including the Unicode encoding, script, character type, number, and name. As the same character name sometimes appears in different scripts, when designing the SI algorithm, the relationship between the character encoding and the script was carefully taken into consideration.

```
# ================================================

3041..3096    ; Hiragana # Lo  [86] HIRAGANA LETTER SMALL A..HIRAGANA LETTER SMALL KE
309D..309E    ; Hiragana # Lm   [2] HIRAGANA ITERATION MARK..HIRAGANA VOICED ITERATION MARK
309F          ; Hiragana # Lo       HIRAGANA DIGRAPH YORI
1B001..1B11F  ; Hiragana # Lo [287] HIRAGANA LETTER ARCHAIC YE..HIRAGANA LETTER ARCHAIC WU
1B132         ; Hiragana # Lo       HIRAGANA LETTER SMALL KO
1B150..1B152  ; Hiragana # Lo   [3] HIRAGANA LETTER SMALL WI..HIRAGANA LETTER SMALL WO
1F200         ; Hiragana # So       SQUARE HIRAGANA HOKA

# Total code points: 381

# ================================================

30A1..30FA    ; Katakana # Lo  [90] KATAKANA LETTER SMALL A..KATAKANA LETTER VO
30FD..30FE    ; Katakana # Lm   [2] KATAKANA ITERATION MARK..KATAKANA VOICED ITERATION MARK
30FF          ; Katakana # Lo       KATAKANA DIGRAPH KOTO
31F0..31FF    ; Katakana # Lo  [16] KATAKANA LETTER SMALL KU..KATAKANA LETTER SMALL RO
32D0..32FE    ; Katakana # So  [47] CIRCLED KATAKANA A..CIRCLED KATAKANA WO
3300..3357    ; Katakana # So  [88] SQUARE APAATO..SQUARE WATTO
FF66..FF6F    ; Katakana # Lo  [10] HALFWIDTH KATAKANA LETTER WO..HALFWIDTH KATAKANA LETTER SMALL TU
FF71..FF9D    ; Katakana # Lo  [45] HALFWIDTH KATAKANA LETTER A..HALFWIDTH KATAKANA LETTER N
1AFF0..1AFF3  ; Katakana # Lm   [4] KATAKANA LETTER MINNAN TONE-2..KATAKANA LETTER MINNAN TONE-5
1AFF5..1AFFB  ; Katakana # Lm   [7] KATAKANA LETTER MINNAN TONE-7..KATAKANA LETTER MINNAN NASALIZED TONE-5
1AFFD..1AFFE  ; Katakana # Lm   [2] KATAKANA LETTER MINNAN NASALIZED TONE-7..KATAKANA LETTER MINNAN NASALIZED TONE-8
1B000         ; Katakana # Lo       KATAKANA LETTER ARCHAIC E
1B120..1B122  ; Katakana # Lo   [3] KATAKANA LETTER ARCHAIC YI..KATAKANA LETTER ARCHAIC WU
1B155         ; Katakana # Lo       KATAKANA LETTER SMALL KO
1B164..1B167  ; Katakana # Lo   [4] KATAKANA LETTER SMALL WI..KATAKANA LETTER SMALL N

# Total code points: 321

# ================================================

02EA..02EB    ; Bopomofo # Sk   [2] MODIFIER LETTER YIN DEPARTING TONE MARK..MODIFIER LETTER YANG DEPARTING TONE MARK
3105..312F    ; Bopomofo # Lo  [43] BOPOMOFO LETTER B..BOPOMOFO LETTER NN
31A0..31BF    ; Bopomofo # Lo  [32] BOPOMOFO LETTER BU..BOPOMOFO LETTER AH

# Total code points: 77
```

Figure 1. Data for part of the script.

 doi:10.20944/preprints202512.1082.v1

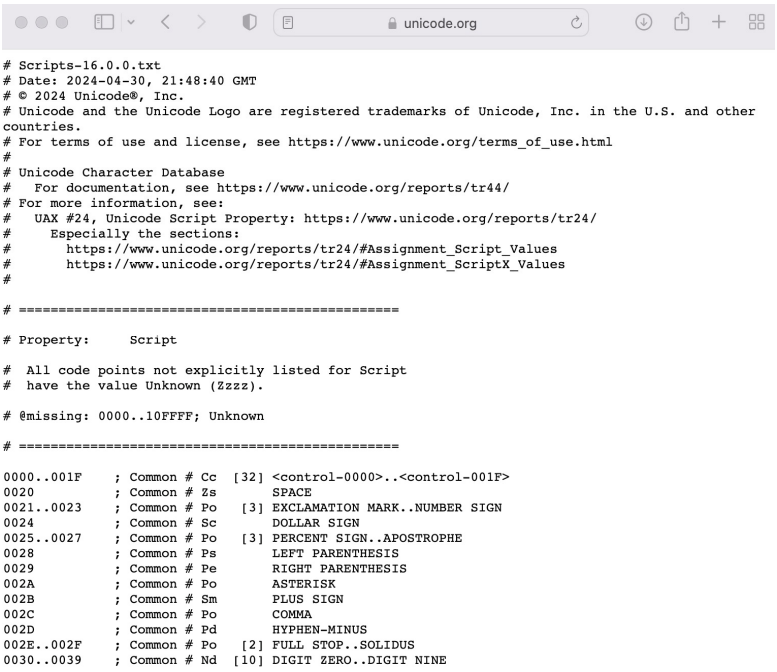

Figure 2. Part of the data for a common script.

2.2. Constructing Regular Expressions for Scripts

There are four types of Unicode encoding data in the Unicode character database, as shown in Table 1, where each 'X' represents a single hexadecimal digit. Among these, Type 1 and Type 3 represent Unicode encodings with only one form, while Type 2 and Type 4 represent multiple consecutive Unicode encodings of the same type.

Table 1. The format of character encodings.

Type 1	Type 2	Type 3	Type 4
XXXX	XXXX ..XXXX	XXXXX	XXXXX ..XXXXX

In the preliminary study, to identify all scripts on Unicode websites while ensuring the integrity of the information in the SI results, the Unicode character database was analyzed and two types of regular expressions were constructed. The first was the common regular expression (CRE), the purpose of which is to identify several characters and symbols shared by scripts that appear in the text. This regular expression was constructed by analyzing the data in the Unicode character table classified as 'common'. The CRE are shown in Table A1 in Appendix A; all content in the table (other than that in bold) are CRE.

The second type of regular expression is the script regular expression (SRE). This type was constructed by analyzing the Unicode codes of characters in scripts other than those labeled as 'common'; all SREs are shown in Table A2 in Appendix A. In addition to characters, text may also contain various symbols. To ensure information integrity, these symbols must also be retained in the SI results. However, integrity cannot be guaranteed when only using SREs. Therefore, the SRE for each script was combined with the CRE to construct another regular expression, called the Mixed Common–Script Regular Expression (MCSRE). An example of a constructed MCSRE is shown in Table A1 in Appendix A; due to spatial limitations, the MCSREs of all scripts are not listed here. Using the CRE and SREs provided in Appendix A, the MCSRE for each script was constructed. These MCSREs were then used alongside the CRE and SREs in the SI algorithm designed in this study.

 doi:10.20944/preprints202512.1082.v1

2.3. Script Identification Dataset

At present, there is no open-source dataset designed specifically for SI. The Leipzig Corpora Collection's source texts were downloaded from news or other websites, and doesn't contain sentences in various foreign languages. The texts are organized by language, with sentences as the basic unit, and datasets of different sizes are provided for each language [16]. It is very useful for building LI datasets as it includes most of the textual resources currently available on the Internet. In a preliminary study, data covering 273 languages were downloaded from the Leipzig Corpora Collection, and 261 languages that are written using only one script were selected. Among these 261 corpora, 208 contained at least 10,000 sentences each, while the other 53 contained fewer than 10,000 sentences [17,18]. When constructing the SI dataset, we first determined which script each text was written in and then labelled the text with that script. The details of the SI dataset are presented in Tables 2–4, in which language ISO codes are used to list the languages considered in this study. Through language encoding, the corresponding language name and country can be searched for on the relevant website [19,20].

Table 2. Scripts used in multiple languages.

Script	Language
Arabic	ara, arz, ckb, fas, glk, kur, pes, pnb, prs, pus, skr, snd, uig, urd, mzn
Han	wuu, zho, gan, cmn, jpn
Cyrillic	bak, bel, bew, bua, bul, che, chv, kaz, kbd, khk, kir, koi, kom, krc, mhr, mkd, mkw, mon, mrj, myv, oss, rue, rus, sah, srp, tat, tgk, tyv, udm, ukr, uzn-uz
Devanagari	hin, mar, nep, new, san, bih
Bengali	asm, ben, bpy
Ethiopic	amh, tir
Georgian	kat, xmf
Greek	ell, pnt
Hebrew	heb, ydd, yid
Kannada	kan, tcy
Latin	ace, ach, afr, aka, als, anw, arg, ast, aym, aze, azj, bam, ban, bar, bcl, bik, bjn, bos, bre, bug, cat, cdo, ceb, ces, cos, csb, cym, dan, deu, diq, dsb, dyu, ekk, emk, eml, eng, epo, est, eus, ewe, ext, fao, fin, fon, fra, frr, fry, fuc, ful, gle, glg, glv, gom, grn, gsw, hat, hau, hbs, hif, hil, hrv, hsb, hun, ibb, ibo, ido, ile, ilo, ina, ind, isl, ita, jav, kab, kal, kbp, kea, kik, kin, kon, ksh, lad, lat, lav, lij, lim, lin, lit, lmo, ltz, lug, lup, lus, lvs, mad_id, min, mlg, mlt, mri, msa, mwl, nan, nap-tara, nav, nbl, ndo, nds, ngl, nld, nob, nno, nor, nso, nya, nyn, oci-fr, orm, pag, pam, pap, pcm, pfl, plt, pms, pol, por, que, roh, rom, ron, run, scn, sco, she, sgs, slk, slv, sme, smi, sna-zw, snk, som, sot-za, spa, sqi, srd, ssw-za, suk, sun, sus, swa, swe, swh, szl, tem, tgl, tiv, tsn, tso, tuk, tum, tur, uzb, vec, ven-za, vie, vls, vol, vro, war, wln, wol, xho-za, yor, zea, zha, zsm, zul-za

Table 3. Scripts that are used in only one language.

Script	Language	Script	Language
Armenian	hye	Sinhala	sin
Gujarati	guj	Tamil	tam
Gurmukhi	pan	Telugu	tel
Hangul	kor	Thaana	div
Khmer	khm	Thai	tha
Lao	lao	Tibetan	bod
Oria	ori		

 doi:10.20944/preprints202512.1082.v1

Table 4. Statistics of scripts used in different languages.

Script	Language Number	Sentence Number	Script	Language Number	Sentence Number
Latin	178	1,387,071	Gujarati	1	10,000
Cyrillic	31	299,245	Gurmukhi	1	10,000
Arabic	15	140,088	Hangul	1	10,000
Devanagari	6	60,000	Lao	1	10,000
Han	5	49,898	Oria	1	10,000
Bengali	3	30,000	Sinhala	1	10,000
Hebrew	2	30,000	Tamil	1	10,000
Georgian	2	20,000	Telugu	1	10,000
Kannada	2	20,000	Thaana	1	10,000
Greek	2	11,564	Thai	1	10,000
Ethiopic	2	11,379	Tibetan	1	7525
Armenian	1	10,000	Khmer	1	1773

3. Script Identification Algorithm

3.1. Script Identification Algorithms in Previous Research

To identify scripts and ensure the integrity of the SI results, the character database provided on the Unicode website was analyzed to construct CRE and MCSREs, on the basis of which the improved SI (ISI) algorithm was designed [13]. The ISI algorithm can recognize all scripts currently recorded on the Unicode website and has good scalability. Its workflow follows the flow diagram shown in Figure 3, comprising the following steps:

1) The text is matched with CREs to extract the parts of the text for which the script tag is «common', and the collective length of these parts is calculated. The matched content consists of symbols, numbers, and other elements shared by several scripts.
2) The text is matched with the MCSRE of each script. When analyzing the matching results for each script, one of the following two situations will occur:
    a) If the length of the MCSRE match result is equal to the length of the CRE match result, the text does not contain content in the current script. Then, MCSRE matching with the next script will be performed on the text.
    b) If the length of the MCSRE match result is not equal to the length of the CRE match result, the text contains content in the current script. As each MCSRE includes space encoding, if the text contains multiple scripts, some matches will have consecutive spaces. After replacing consecutive spaces in the match results with a single space and removing spaces at the beginning and end of the text, the script and match results are added to the results dictionary. Then, MCSRE matching with the next script will be performed on the text.
3) After MCSRE matching is completed for the text and all scripts, a results dictionary is returned, which provides the contents in different scripts and the associated script names.

3.2. Improving the Time Efficiency of Script Identification

In the proposed algorithm, the time efficiency of SI is improved through implementation of the following four methods.

1) The text may contain content in multiple scripts. As the text length is fixed and the sum of the lengths of the content in each script is equal to the total text length, if the total length of the recognized script content equals the length of the text during the SI process, there is no need to analyze the remaining scripts and the process should be terminated immediately.
2) On the Internet, the proportion of text written in different languages varies. Based on the correspondence between languages and scripts, the proportion of commonly used scripts in

 doi:10.20944/preprints202512.1082.v1

online text information can be deduced. Based on this, in an SI algorithm, scripts that appear more frequently on the Internet should be given higher priority. Some websites [14] provide statistics on Internet content in different languages, but there are no corresponding corpora. Therefore, in this study, the constructed script recognition corpus was analyzed and the weight of a given script was set according to the number of sentences contained in the associated corpus. Table 4 shows the number of sentences in each script corpus. When constructing SREs and MCSREs, regular expression sequences were sorted according to the number of sentences in descending order. When using the SI algorithm to identify scripts, the text should be matched with the regular expression of the script with the highest weight first, then analyzed with the script with the next highest weight, and so on. Ideally, a text will contain only one type of script content; if this is the case, the target script can be quickly located based on above matching pattern. Once located, there is no need to continue analyzing the remaining scripts. This can improve the time efficiency of SI.

3) During the SI process, determining whether the text contains content written in a specific script is a key operation. If there are many scripts that need to be identified, this operation must be performed multiple times. In preliminary research, MCSRE matching operations were used to analyze whether the text contains content written in the current script. As SREs are shorter than MCSREs, we chose to use SRE matching operations to improve the matching speed. Each time SRE matching is performed, if a match result is returned, the corresponding script and matching result are added to the SI result dictionary.
4) The results obtained from SRE matching do not include punctuation marks, numbers, spaces, and certain characters which are common to multiple scripts. If these symbols are not used for SI, the completeness of the text information cannot be ensured, which affects subsequent text processing results. To ensure the integrity of the information in the SI results, MCSREs were constructed in the preliminary stage of research. This type of regular expression includes not only symbols specific to each script, but also various symbols shared by multiple scripts. The MCSRE matching results can ensure the integrity of the SI results. For sentences containing multiple scripts, where the sentence structure is relatively complex, executing MCSRE matching may result in some symbols that belong to other scripts being included in the results. Therefore, in this study, based on the results obtained through SRE matching, the indices of the first and last characters of the result are first located in the original text. Then, according to these two indices, the text segments where the script content appears are obtained. Finally, the text segments are matched with the MCSREs of the corresponding scripts and consecutive spaces are replaced with a single space, resulting in the final script content.

3.3. High-Time-Efficiency Script Identification Algorithm

In this study, the above four methods are combined to improve the execution efficiency of an SI algorithm based on the ISI algorithm proposed in previous research [13], and the high-time-efficiency script identification (HTESI) algorithm is proposed. In the preliminary study, two types of regular expressions—CRE and MCSREs—were considered to analyze the SI process. Three regular expressions for matching are used in this study, as follows:

1) CRE matching reveals characters shared by multiple scripts in the text, including punctuation, spaces, numbers, characters and other symbols;
2) SRE matching is used to obtain the characters in the text that are used only by the current script. If the match length is non-zero, this indicates that the text contains content written in the current script. Otherwise, the text does not contain content written in the current script;
3) MCSRE matching is performed after determining the type of script and matching result appearing in the text. Each script identified in the SI results is matched with the original text using MCSRE to obtain complete recognition information.

 doi:10.20944/preprints202512.1082.v1

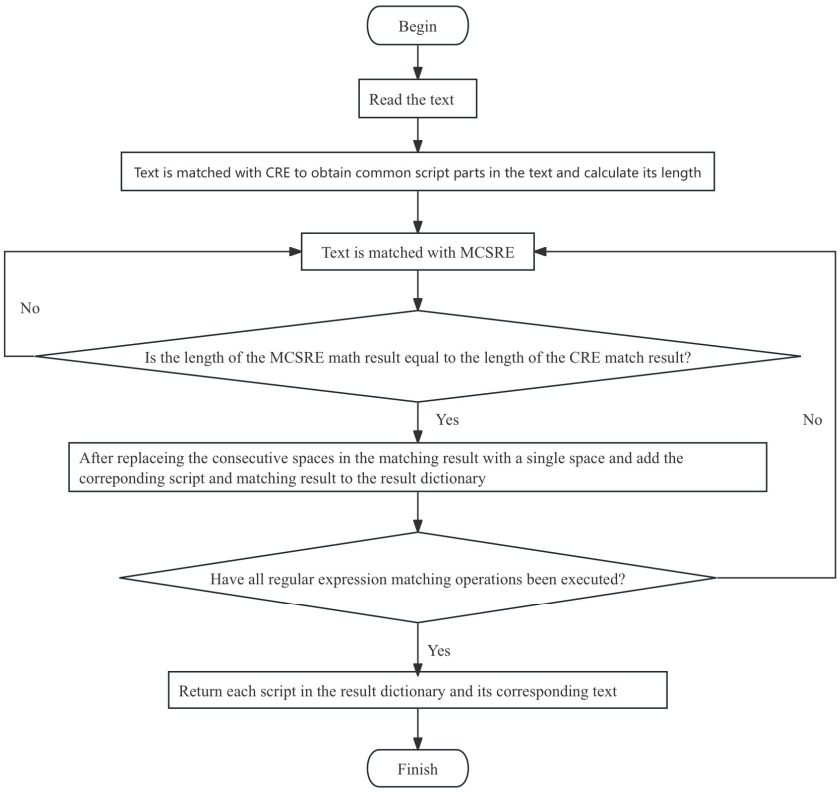

Figure 3. Flowchart of the ISI algorithm.

The workflow of the HTESI algorithm follows the flow diagram shown in Figure 4, comprising the following steps:

1) The text length L_t is calculated and the variable L_{sum} is defined to calculate the total length of the SI result.
2) This text is matched with CREs to extract the parts of the text where the script tag is 'common' and the length L_c of these texts is calculated. At this point, $L_{sum} = L_c$.
3) The text is matched with the SRE of each script. When analyzing the matching results for each script, one of the following two situations will occur:
4) If the matching result length L_m is equal to zero, execute the SRE matching operation of the next script.
5) If the match result l_m is not zero, add the corresponding script to the SI result dictionary and calculate $L_{sum} = L_{sum} + L_m$. Then, check if L_{sum} is equal to L_t: if so, stop the SI; otherwise, perform the matching operation for the next script's SRE.
6) After completing SI, for each script and matching result in the SI dictionary, the start and end characters of the matching result are located in the original text to determine the text segment in which the script appears. Then, a matching operation is performed between this text segment and the corresponding script's MCSRE, consecutive spaces are replaced with a single space, and the matching results—along with the corresponding script—are added to the final SI result dictionary.

 doi:10.20944/preprints202512.1082.v1

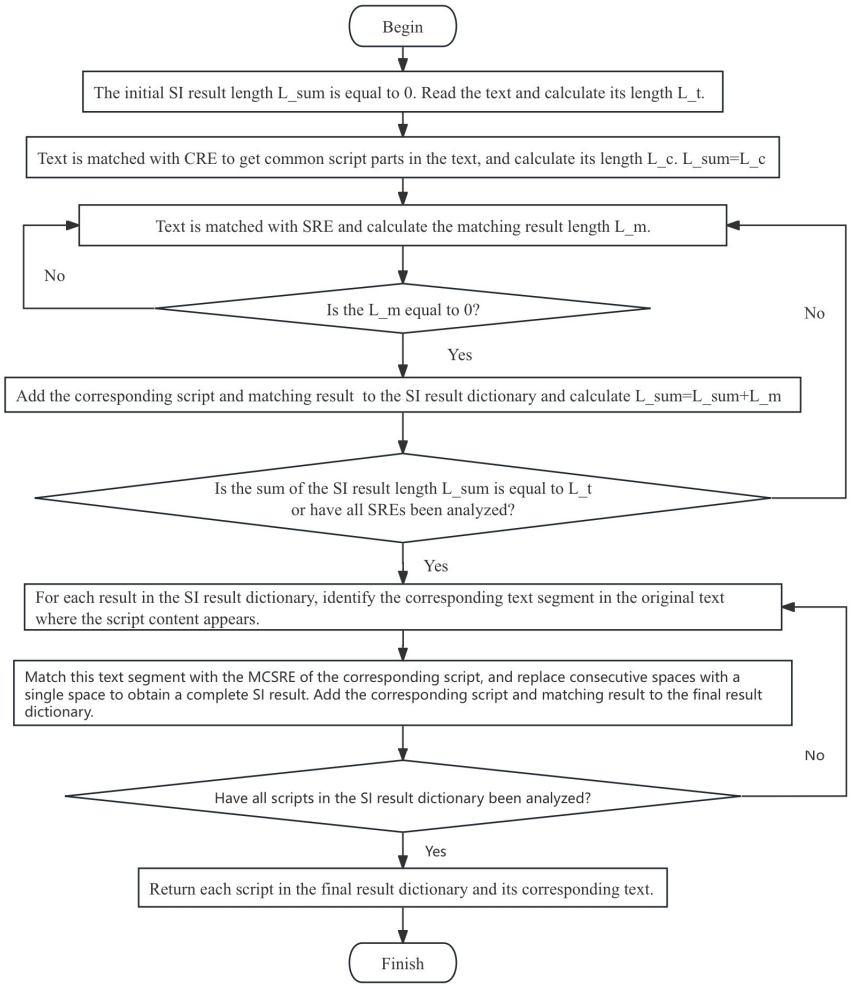

Figure 4. Flowchart of the high-time-efficiency script identification (HTESI) algorithm.

3.4. HTESI Example

To illustrate the HTESI process, we selected the following sentence containing multiple scripts from the corpus: "Bloomberg News со ссылкой на проект заявления G7 по итогам заседания." The HTESI process for this sentence is detailed in Table 5. In this study, weights were set based on the number of sentences in each script corpus and the MSE and MCSRE sequences were sorted in descending order of weight. Most content in the corpora was in Latin script, followed by Slavic languages. Therefore, in this case, the text was first matched with the Latin SREs. As the sum of the SI results after matching was less than the length of the original text, the text was then matched with the SREs corresponding to the second-ranked Slavic language in the SRE sequence. When the sum of the SI results after matching was found to be equal to the length of the original text, there was no need to continue analyzing the remaining scripts in the SRE sequence and the SI process stopped. After SI was completed, the segment of the original text where the script content appears was located for each result, which was then matched using MCSREs to form a complete SI result.

Table 5. Example of the HTESI process.

ISI Process	The Result of Each Step
Calculate the length of the text; text-length = 69. The initial SI result value is equal to 0, represented as SI-result-length = 0.	Text-length = 69 SI-result-length = 0

 doi:10.20944/preprints202512.1082.v1

Use CREs to identify the parts of text in a common script and calculate their length.	Common script part is [’7’, ’], with Common-length = 12. SI-result-length = Common-length = 12
Match the text to the SREs of each script separately and analyze the matching results after each execution of an MCSRE. Initially, the text was matched with the Latin SRE. The length of the match result is equal to 14, and SI-result-length = 26. As SI-result-length< text-length, regular expression matching with the next script, Cyrillic, was then performed.	Latin = [’B’, ’l’, ’o’, ’o’, ’m’, ’b’, ’e’, ’r’, ’g’, ’N’, ’e’, ’w’, ’s’, ’G’], Current-result-length = 14 SI-result-length = Current-result-length + Common-length = 14 + 12 = 26
The length of the match result is equal to 43, and SI-result-length = 69. As SI-result-length is equal to the text-length, the SI process is stopped.	Cyrillic = [’, ’]} Current-result-length = 43 Script-result-length = Current-result-length + Common-length = 43 + 26 = 69
After SI is completed, for each result, identify the corresponding text segment in the original text where the script content appears. Then, match this text segment with the MCSRE of the corresponding script and replace consecutive spaces with a single space to obtain a complete SI result.	The corresponding Latin text segment is Latin = “Bloomberg News со ссылкой на проект заявления G7”. Performing MCSRE matching and replacing consecutive spaces with a single space, the result is “Bloomberg News G7”. The corresponding Cyrillic text segment is Cyrillic = “ со ссылкой на проект заявления G7 по итогам заседания”. Performing MCSRE matching and replacing consecutive spaces with a single space, the result is “ссылкой на проект заявления 7 по итогам заседания”.
Return the SI result.	{’Latin’: [Bloomberg News G7], ’Cyrillic’: [ссылкой на проект заявления 7 по итогам заседания]}

4. Results

To verify the feasibility of the algorithm proposed in this study, training and testing experiments were conducted using the SI dataset mentioned in Section 2.3. The ratio of the training set to the test set was 9:1. To verify the proposed SI algorithm’s efficiency, we designed three different experiments and performed comparative experiments using the ISI algorithm proposed in previous research [13]. The experimental environment of this study was a MacBook Pro with an Apple M1 chip,8GB of memory and the program was written in Python 3.11.5 using the Spyder 5.4.3 IDE.

SI is a text classification task, in which each entry in the text classification corpus must have a label and, so, the text classification algorithm must return a label when predicting the text. When evaluating the performance of an SI algorithm, it is used to determine which script is contained in the text; therefore, the number of words separated by a space was counted, and the script with the most words was selected as the main script of the text. As Han scripts encode more information in a shorter number of characters than, for example, Latin scripts, the Chinese and Japanese languages in the Han group are written without spaces between the words. Thus, for the Han group, the length of the matching result was considered, except for the spaces.

The performance of the text classification algorithms was evaluated using Precision, Recall, and F-score. This study used the Micro P, Micro R, and Micro F_1 metrics to evaluate the performance of the SI algorithm, as shown in Equations (1)–(3), in which C represents all the scripts in the SI data set; |C| denotes the number of scripts in the SI data set; and the meanings of TP, FP, and FN are given in Table 6.

 doi:10.20944/preprints202512.1082.v1

Table 6. Definitions related to evaluation indicators.

	The number of texts that belong to the script	The number of texts that do not belong to the script
The number of texts that the SI algorithm determines to belong to the script	True positive (TP)	False positive (FP)
The number of texts that the SI algorithm determines to not belong to the script	False negative (FN)	True negative (TN)

$$\text{MicroP} = \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \text{TP}_i + \sum_{i=1}^{|C|} \text{FP}_i}, \quad (1)$$

$$\text{MicroR} = \frac{\sum_{i=1}^{|C|} \text{TP}_i}{\sum_{i=1}^{|C|} \text{TP}_i + \sum_{i=1}^{|C|} \text{FN}_i}, \quad (2)$$

$$\text{MicroF}_1 = \frac{2 \times \text{MicroP} \times \text{MicroR}}{\text{MicroP} + \text{MicroR}}. \quad (3)$$

4.1. Impact of SRE Matching on SI

In the preliminary study, when determining whether a text contains a certain type of script, the text was matched with the corresponding MCSRE of that script. The purpose of this was to ensure the integrity of the SI result. The main operation in SI is to match text with the regular expressions of each script. In this study, SREs were initially used for matching to reduce the matching time, as they are simpler than MCSREs.

To verify this method's validity, an algorithm was designed and labeled SI-1. The workflow of SI-1 is almost the same as that of the ISI algorithm, with the difference being that when determining whether the text contains a certain script, an SRE is used instead of an MCSRE, and segments which contain a certain script are determined after identifying different scripts in the text and then matched with the MCSRE of the corresponding script to obtain the SI results. To better understand the experimental process, Figure 5 shows the flowchart of SI-1.

The results of a comparative experiment between SI-1 and ISI are shown in Table 7. These results show that using SREs improved the matching speed; in particular, the training and testing times of the newly proposed method were reduced by 8.61- and 8.56-fold than those of previous study, respectively, respectively, while the Micro F1 score for script identification was slightly higher than those reported in our previous study.

Table 7. SI experiment results.

Algorithm	Experiment	Micro P	Micro R	Micro F_1	Time (second)
ISI	train	0.9926	0.9926	0.9926	452.14
	test	0.9924	0.9924	0.9924	113.01
SI-1	train	0.9930	0.9930	0.9930	70.15
	test	0.9929	0.9929	0.9929	17.29
SI-2	train	0.9930	0.9930	0.9930	54.20
	test	0.9929	0.9929	0.9929	13.72
HTESI	train	0.9930	0.9930	0.9930	52.47
	test	0.9929	0.9929	0.9929	13.19

 doi:10.20944/preprints202512.1082.v1

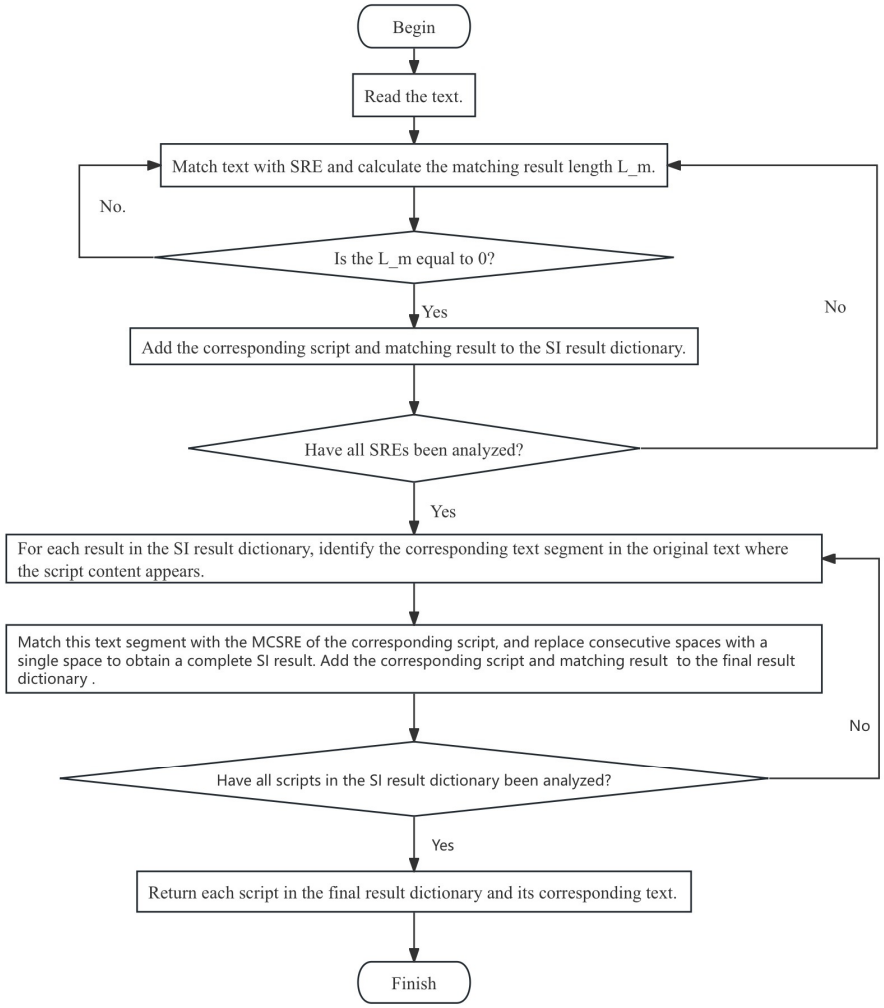

Figure 5. Flowchart of the SI-1 algorithm.

4.2. The Impact of Calculating Match Length on SI Algorithm Performance

The sum of the lengths of all script texts that appear in the text is equal to the total text length—a potential advantage that was not considered in previous research. If the sum of the identified SI results is equal to the text's length, there is no need to perform further regular expression matching operations between the remaining scripts and the text. In this research, this advantage is leveraged to improve the proposed algorithm's time efficiency. To verify the validity of this method, an algorithm was designed, labeled SI-2. The workflow of the algorithm is the same as that of the algorithm proposed in this research (shown in Figure 4); the main difference is that the frequency with which each script appears in the corpora is not considered—the order of the regular expressions for each script used in the comparative experiment is instead consistent with the order of scripts on the official Unicode website.

The results of the comparative experiment between SI-1, SI-2, and ISI are shown in Table 7, showing that the algorithm's time efficiency was further improved by considering the length of the SI results, the training and testing times were reduced by 8.34- and 8.23-fold than those of previous study respectively, respectively, without reducing the algorithm's performance.

 doi:10.20944/preprints202512.1082.v1

4.3. The Impact of the Frequency of Different Scripts on SI Performance

On the Internet, the proportion of text written in different scripts varies. When designing an SI algorithm, if the regular expressions for each script are sorted in descending order based on their frequency on the Internet, it is likely that fewer pattern analyses will need to be performed, and the most-used scripts can be identified rapidly; in this way, the algorithm's time efficiency can be further improved. As there is no relevant open-source corpus available for this purpose, in this study the weights were set based on the number of sentences in each script corpus (as detailed in Table 4), and the SREs and MCSREs were sorted according to these weights. After ranking the SREs and MCSREs in this way, the HTESI algorithm proposed in this study could be implemented; its process is shown in Figure 4. The results of a comparison of the relevant experimental results are shown in Table 7, the training and testing times were reduced by 8.61- and 8.56-fold than those of previous study, respectively. It is indicating that the SI algorithm's time efficiency was further improved through weighting by the prevalence of different scripts, without any adverse effects on the algorithm's performance.

5. Conclusions

SI is the first step in most multilingual text processing tasks. To improve the time efficiency of the SI algorithm, it was optimized in the following three ways in this study:

1) The main operation in the SI process is to determine whether a certain type of script is present in the text. In previous research, to provide complete SI results for subsequent text processing, each script-specific character encoding and the character encodings shared among scripts were merged to construct an MCSRE for each script. As the content represented by this regular expression is relatively extensive, performing regular expression matching requires a significant amount of time. To reduce the time needed for this process, we constructed SREs for each script using only script-specific character encodings. As these SREs carry far less information than MCSREs, performing SRE matching requires less time. Therefore, in this study, SREs are used to determine whether certain scripts are present in the text. If present, MCSREs are then used to obtain complete SI results.
2) In preliminary research, to determine which script is present in a text, the text was matched against the regular expressions of various scripts using regular expression matching. This study takes advantage of the fact that the sum of the lengths of content written in different scripts within a text is equal to the total length of the text by calculating the sum of the lengths of each recognition result during the SI process. If this sum is equal to the length of the text, there is no need to analyze the remaining scripts and the SI process is stopped.
3) The frequency of text in different languages on the Internet varies; similarly, the frequency of content written in different scripts also varies. Therefore, when designing an SI algorithm, it is preferable to first analyze those scripts that appear more frequently to identify them rapidly. This study also takes advantage of this practical situation: when designing the list of regular expressions, the expressions were sorted in descending order according to the frequency of content written in each script in the corpus. This allows the SI process to determine content in different scripts present in the text more quickly.

The experimental results show that, after applying these optimization strategies, the time efficiency of the resulting HTESI algorithm noticeably improved without reducing the identification rate, thus demonstrating the effectiveness of the optimization methods proposed in this study.

Author Contributions: M.Q. designed the algorithm, analyzed the data, and wrote the paper. W.S. supervised the writing and reviewed the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (62476234).

 doi:10.20944/preprints202512.1082.v1

Data Availability Statement: All experimental data in this study were obtained from the Leipzig Corpora Collection: https://cls.corpora.uni-leipzig.de/en (accessed on 2 June 2024) and https://www.unicode.org/Public/UNIDATA/Scripts.txt.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SI	Script identification
LI	Language identification
CRE	Common script regular expression
SRE	Script regular expression
MCSRE	The regular expression formed by CRE and SRE
HTESI	Higth time efficiency script identification

Appendix A

Table A1. An example of a constructed MCSRE for Hiragana script.

[\\u3041-\\u3096\\u309D-\\u309E\\u309F\\U0001B001-\\U0001B11F\\U0001B132\\U0001B150-\\U0001B152\\U0001F200\\u0000-\\u001F\\u0020\\u0021-\\u0023\\u0024\\u0025-
\\u0027\\u0028\\u0029\\u002A\\u002B\\u002C\\u002D\\u002E-\\u002F\\u0030-
\\u0039\\u003A-\\u003B\\u003C-\\u003E\\u003F-
\\u0040\\u005B\\u005C\\u005D\\u005E\\u005F\\u0060\\u007B\\u007C\\u007D\\u007E\
\\u007F-\\u009F\\u00A0\\u00A1\\u00A2-
\\u00A5\\u00A6\\u00A7\\u00A8\\u00A9\\u00AB\\u00AC\\u00AD\\u00AE\\u00AF\\u0
0B0\\u00B1\\u00B2-\\u00B3\\u00B4\\u00B5\\u00B6-
\\u00B7\\u00B8\\u00B9\\u00BB\\u00BC-\\u00BE\\u00BF\\u00D7\\u00F7\\u02B9-
\\u02C1\\u02C2-\\u02C5\\u02C6-\\u02D1\\u02D2-\\u02DF\\u02E5-
\\u02E9\\u02EC\\u02ED\\u02EE\\u02EF-
\\u02FF\\u0374\\u037E\\u0385\\u0387\\u0605\\u060C\\u061B\\u061F\\u0640\\u06DD\
\\u08E2\\u0964-\\u0965\\u0E3F\\u0FD5-\\u0FD8\\u10FB\\u16EB-\\u16ED\\u1735-
\\u1736\\u1802-\\u1803\\u1805\\u1CD3\\u1CE1\\u1CE9-\\u1CEC\\u1CEE-
\\u1CF3\\u1CF5-\\u1CF6\\u1CF7\\u1CFA\\u2000-\\u200A\\u200B\\u200E-
\\u200F\\u2010-\\u2015\\u2016-\\u2017\\u2018\\u2019\\u201A\\u201B-
\\u201C\\u201D\\u201E\\u201F\\u2020-\\u2027\\u2028\\u2029\\u202A-
\\u202E\\u202F\\u2030-\\u2038\\u2039\\u203A\\u203B-\\u203E\\u203F-\\u2040\\u2041-
\\u2043\\u2044\\u2045\\u2046\\u2047-\\u2051\\u2052\\u2053\\u2054\\u2055-
\\u205E\\u205F\\u2060-\\u2064\\u2066-\\u206F\\u2070\\u2074-\\u2079\\u207A-
\\u207C\\u207D\\u207E\\u2080-\\u2089\\u208A-\\u208C\\u208D\\u208E\\u20A0-
\\u20C0\\u2100-\\u2101\\u2102\\u2103-\\u2106\\u2107\\u2108-\\u2109\\u210A-
\\u2113\\u2114\\u2115\\u2116-\\u2117\\u2118\\u2119-\\u211D\\u211E-
\\u2123\\u2124\\u2125\\u2127\\u2128\\u2129\\u212C-\\u212D\\u212E\\u212F-
\\u2131\\u2133-\\u2134\\u2135-\\u2138\\u2139\\u213A-\\u213B\\u213C-
\\u213F\\u2140-\\u2144\\u2145-\\u2149\\u214A\\u214B\\u214C-
\\u214D\\u214F\\u2150-\\u215F\\u2189\\u218A-\\u218B\\u2190-\\u2194\\u2195-
\\u2199\\u219A-\\u219B\\u219C-\\u219F\\u21A0\\u21A1-\\u21A2\\u21A3\\u21A4-
\\u21A5\\u21A6\\u21A7-\\u21AD\\u21AE\\u21AF-\\u21CD\\u21CE-\\u21CF\\u21D0-
\\u21D1\\u21D2\\u21D3\\u21D4\\u21D5-\\u21F3\\u21F4-\\u22FF\\u2300-
\\u2307\\u2308\\u2309\\u230A\\u230B\\u230C-\\u231F\\u2320-\\u2321\\u2322-
\\u2328\\u2329\\u232A\\u232B-\\u237B\\u237C\\u237D-\\u239A\\u239B-
\\u23B3\\u23B4-\\u23DB\\u23DC-\\u23E1\\u23E2-\\u2429\\u2440-\\u244A\\u2460-
\\u249B\\u249C-\\u24E9\\u24EA-\\u24FF\\u2500-\\u25B6\\u25B7\\u25B8-

 doi:10.20944/preprints202512.1082.v1

\\u25C0\\u25C1\\u25C2-\\u25F7\\u25F8-\\u25FF\\u2600-\\u266E\\u266F\\u2670-
\\u2767\\u2768\\u2769\\u276A\\u276B\\u276C\\u276D\\u276E\\u276F\\u2770\\u2771\\
u2772\\u2773\\u2774\\u2775\\u2776-\\u2793\\u2794-\\u27BF\\u27C0-
\\u27C4\\u27C5\\u27C6\\u27C7-
\\u27E5\\u27E6\\u27E7\\u27E8\\u27E9\\u27EA\\u27EB\\u27EC\\u27ED\\u27EE\\u27E
F\\u27F0-\\u27FF\\u2900-
\\u2982\\u2983\\u2984\\u2985\\u2986\\u2987\\u2988\\u2989\\u298A\\u298B\\u298C\\
u298D\\u298E\\u298F\\u2990\\u2991\\u2992\\u2993\\u2994\\u2995\\u2996\\u2997\\u29
98\\u2999-\\u29D7\\u29D8\\u29D9\\u29DA\\u29DB\\u29DC-
\\u29FB\\u29FC\\u29FD\\u29FE-\\u2AFF\\u2B00-\\u2B2F\\u2B30-\\u2B44\\u2B45-
\\u2B46\\u2B47-\\u2B4C\\u2B4D-\\u2B73\\u2B76-\\u2B95\\u2B97-\\u2BFF\\u2E00-
\\u2E01\\u2E02\\u2E03\\u2E04\\u2E05\\u2E06-
\\u2E08\\u2E09\\u2E0A\\u2E0B\\u2E0C\\u2E0D\\u2E0E-\\u2E16\\u2E17\\u2E18-
\\u2E19\\u2E1A\\u2E1B\\u2E1C\\u2E1D\\u2E1E-
\\u2E1F\\u2E20\\u2E21\\u2E22\\u2E23\\u2E24\\u2E25\\u2E26\\u2E27\\u2E28\\u2E29\
\u2E2A-\\u2E2E\\u2E2F\\u2E30-\\u2E39\\u2E3A-\\u2E3B\\u2E3C-
\\u2E3F\\u2E40\\u2E41\\u2E42\\u2E43-\\u2E4F\\u2E50-\\u2E51\\u2E52-
\\u2E54\\u2E55\\u2E56\\u2E57\\u2E58\\u2E59\\u2E5A\\u2E5B\\u2E5C\\u2E5D\\u2FF0
-\\u2FFF\\u3000\\u3001-
\\u3003\\u3004\\u3006\\u3008\\u3009\\u300A\\u300B\\u300C\\u300D\\u300E\\u300F\\
u3010\\u3011\\u3012-
\\u3013\\u3014\\u3015\\u3016\\u3017\\u3018\\u3019\\u301A\\u301B\\u301C\\u301D\\
u301E-\\u301F\\u3020\\u3030\\u3031-\\u3035\\u3036-\\u3037\\u303C\\u303D\\u303E-
\\u303F\\u309B-\\u309C\\u30A0\\u30FB\\u30FC\\u3190-\\u3191\\u3192-
\\u3195\\u3196-\\u319F\\u31C0-\\u31E5\\u31EF\\u3220-\\u3229\\u322A-
\\u3247\\u3248-\\u324F\\u3250\\u3251-\\u325F\\u327F\\u3280-\\u3289\\u328A-
\\u32B0\\u32B1-\\u32BF\\u32C0-\\u32CF\\u32FF\\u3358-\\u33FF\\u4DC0-
\\u4DFF\\uA700-\\uA716\\uA717-\\uA71F\\uA720-\\uA721\\uA788\\uA789-
\\uA78A\\uA830-\\uA835\\uA836-
\\uA837\\uA838\\uA839\\uA92E\\uA9CF\\uAB5B\\uAB6A-
\\uAB6B\\uFD3E\\uFD3F\\uFE10-\\uFE16\\uFE17\\uFE18\\uFE19\\uFE30\\uFE31-
\\uFE32\\uFE33-
\\uFE34\\uFE35\\uFE36\\uFE37\\uFE38\\uFE39\\uFE3A\\uFE3B\\uFE3C\\uFE3D\\uFE3
E\\uFE3F\\uFE40\\uFE41\\uFE42\\uFE43\\uFE44\\uFE45-
\\uFE46\\uFE47\\uFE48\\uFE49-\\uFE4C\\uFE4D-\\uFE4F\\uFE50-\\uFE52\\uFE54-
\\uFE57\\uFE58\\uFE59\\uFE5A\\uFE5B\\uFE5C\\uFE5D\\uFE5E\\uFE5F-
\\uFE61\\uFE62\\uFE63\\uFE64-\\uFE66\\uFE68\\uFE69\\uFE6A-
\\uFE6B\\uFEFF\\uFF01-\\uFF03\\uFF04\\uFF05-
\\uFF07\\uFF08\\uFF09\\uFF0A\\uFF0B\\uFF0C\\uFF0D\\uFF0E-\\uFF0F\\uFF10-
\\uFF19\\uFF1A-\\uFF1B\\uFF1C-\\uFF1E\\uFF1F-
\\uFF20\\uFF3B\\uFF3C\\uFF3D\\uFF3E\\uFF3F\\uFF40\\uFF5B\\uFF5C\\uFF5D\\uFF5
E\\uFF5F\\uFF60\\uFF61\\uFF62\\uFF63\\uFF64-\\uFF65\\uFF70\\uFF9E-
\\uFF9F\\uFFE0-\\uFFE1\\uFFE2\\uFFE3\\uFFE4\\uFFE5-\\uFFE6\\uFFE8\\uFFE9-
\\uFFEC\\uFFED-\\uFFEE\\uFFF9-\\uFFFB\\uFFFC-\\uFFFD\\U00010100-
\\U00010102\\U00010107-\\U00010133\\U00010137-\\U0001013F\\U00010190-
\\U0001019C\\U000101D0-\\U000101FC\\U000102E1-\\U000102FB\\U0001BCA0-
\\U0001BCA3\\U0001CC00-\\U0001CCEF\\U0001CCF0-\\U0001CCF9\\U0001CD00-
\\U0001CEB3\\U0001CF50-\\U0001CFC3\\U0001D000-\\U0001D0F5\\U0001D100-
\\U0001D126\\U0001D129-\\U0001D164\\U0001D165-\\U0001D166\\U0001D16A-
\\U0001D16C\\U0001D16D-\\U0001D172\\U0001D173-\\U0001D17A\\U0001D183-
\\U0001D184\\U0001D18C-\\U0001D1A9\\U0001D1AE-\\U0001D1EA\\U0001D2C0-
\\U0001D2D3\\U0001D2E0-\\U0001D2F3\\U0001D300-\\U0001D356\\U0001D360-
\\U0001D378\\U0001D400-\\U0001D454\\U0001D456-\\U0001D49C\\U0001D49E-

 doi:10.20944/preprints202512.1082.v1

\\U0001D49F\\U0001D4A2\\U0001D4A5-\\U0001D4A6\\U0001D4A9-
\\U0001D4AC\\U0001D4AE-\\U0001D4B9\\U0001D4BB\\U0001D4BD-
\\U0001D4C3\\U0001D4C5-\\U0001D505\\U0001D507-\\U0001D50A\\U0001D50D-
\\U0001D514\\U0001D516-\\U0001D51C\\U0001D51E-\\U0001D539\\U0001D53B-
\\U0001D53E\\U0001D540-\\U0001D544\\U0001D546\\U0001D54A-
\\U0001D550\\U0001D552-\\U0001D6A5\\U0001D6A8-
\\U0001D6C0\\U0001D6C1\\U0001D6C2-\\U0001D6DA\\U0001D6DB\\U0001D6DC-
\\U0001D6FA\\U0001D6FB\\U0001D6FC-\\U0001D714\\U0001D715\\U0001D716-
\\U0001D734\\U0001D735\\U0001D736-\\U0001D74E\\U0001D74F\\U0001D750-
\\U0001D76E\\U0001D76F\\U0001D770-\\U0001D788\\U0001D789\\U0001D78A-
\\U0001D7A8\\U0001D7A9\\U0001D7AA-\\U0001D7C2\\U0001D7C3\\U0001D7C4-
\\U0001D7CB\\U0001D7CE-\\U0001D7FF\\U0001EC71-
\\U0001ECAB\\U0001ECAC\\U0001ECAD-\\U0001ECAF\\U0001ECB0\\U0001ECB1-
\\U0001ECB4\\U0001ED01-\\U0001ED2D\\U0001ED2E\\U0001ED2F-
\\U0001ED3D\\U0001F000-\\U0001F02B\\U0001F030-\\U0001F093\\U0001F0A0-
\\U0001F0AE\\U0001F0B1-\\U0001F0BF\\U0001F0C1-\\U0001F0CF\\U0001F0D1-
\\U0001F0F5\\U0001F100-\\U0001F10C\\U0001F10D-\\U0001F1AD\\U0001F1E6-
\\U0001F1FF\\U0001F201-\\U0001F202\\U0001F210-\\U0001F23B\\U0001F240-
\\U0001F248\\U0001F250-\\U0001F251\\U0001F260-\\U0001F265\\U0001F300-
\\U0001F3FA\\U0001F3FB-\\U0001F3FF\\U0001F400-\\U0001F6D7\\U0001F6DC-
\\U0001F6EC\\U0001F6F0-\\U0001F6FC\\U0001F700-\\U0001F776\\U0001F77B-
\\U0001F7D9\\U0001F7E0-\\U0001F7EB\\U0001F7F0\\U0001F800-\\U0001F80B\\U0001F810-
\\U0001F847\\U0001F850-\\U0001F859\\U0001F860-\\U0001F887\\U0001F890-
\\U0001F8AD\\U0001F8B0-\\U0001F8BB\\U0001F8C0-\\U0001F8C1\\U0001F900-
\\U0001FA53\\U0001FA60-\\U0001FA6D\\U0001FA70-\\U0001FA7C\\U0001FA80-
\\U0001FA89\\U0001FA8F-\\U0001FAC6\\U0001FACE-\\U0001FADC\\U0001FADF-
\\U0001FAE9\\U0001FAF0-\\U0001FAF8\\U0001FB00-\\U0001FB92\\U0001FB94-
\\U0001FBEF\\U0001FBF0-\\U0001FBF9\\U000E0001\\U000E0020-\\U000E007F]

Table A2. The regular expression for each script (SRE).

No	Script	Unicode Range
1	Latin	[\\u0041-\\u005A\\u0061-\\u007A\\u00AA\\u00BA\\u00C0- \\u00D6\\u00D8-\\u00F6\\u00F8-\\u01BA\\u01BB\\u01BC- \\u01BF\\u01C0-\\u01C3\\u01C4-\\u0293\\u0294\\u0295- \\u02AF\\u02B0-\\u02B8\\u02E0-\\u02E4\\u1D00-\\u1D25\\u1D2C- \\u1D5C\\u1D62-\\u1D65\\u1D6B-\\u1D77\\u1D79-\\u1D9A\\u1D9B- \\u1DBE\\u1E00-\\u1EFF\\u2071\\u207F\\u2090-\\u209C\\u212A- \\u212B\\u2132\\u214E\\u2160-\\u2182\\u2183-\\u2184\\u2185- \\u2188\\u2C60-\\u2C7B\\u2C7C-\\u2C7D\\u2C7E-\\u2C7F\\uA722- \\uA76F\\uA770\\uA771-\\uA787\\uA78B-\\uA78E\\uA78F\\uA790- \\uA7CD\\uA7D0-\\uA7D1\\uA7D3\\uA7D5-\\uA7DC\\uA7F2- \\uA7F4\\uA7F5-\\uA7F6\\uA7F7\\uA7F8-\\uA7F9\\uA7FA\\uA7FB- \\uA7FF\\uAB30-\\uAB5A\\uAB5C-\\uAB5F\\uAB60- \\uAB64\\uAB66-\\uAB68\\uAB69\\uFB00-\\uFB06\\uFF21- \\uFF3A\\uFF41-\\uFF5A\\U00010780-\\U00010785\\U00010787- \\U000107B0\\U000107B2-\\U000107BA\\U0001DF00- \\U0001DF09\\U0001DF0A\\U0001DF0B-\\U0001DF1E\\U0001DF25- \\U0001DF2A]

 doi:10.20944/preprints202512.1082.v1

2	Greek	[\\u0370-\\u0373\\u0375\\u0376-\\u0377\\u037A\\u037B-\\u037D\\u037F\\u0384\\u0386\\u0388-\\u038A\\u038C\\u038E-\\u03A1\\u03A3-\\u03E1\\u03F0-\\u03F5\\u03F6\\u03F7-\\u03FF\\u1D26-\\u1D2A\\u1D5D-\\u1D61\\u1D66-\\u1D6A\\u1DBF\\u1F00-\\u1F15\\u1F18-\\u1F1D\\u1F20-\\u1F45\\u1F48-\\u1F4D\\u1F50-\\u1F57\\u1F59\\u1F5B\\u1F5D\\u1F5F-\\u1F7D\\u1F80-\\u1FB4\\u1FB6-\\u1FBC\\u1FBD\\u1FBE\\u1FBF-\\u1FC1\\u1FC2-\\u1FC4\\u1FC6-\\u1FCC\\u1FCD-\\u1FCF\\u1FD0-\\u1FD3\\u1FD6-\\u1FDB\\u1FDD-\\u1FDF\\u1FE0-\\u1FEC\\u1FED-\\u1FEF\\u1FF2-\\u1FF4\\u1FF6-\\u1FFC\\u1FFD-\\u1FFE\\u2126\\uAB65\\U00010140-\\U00010174\\U00010175-\\U00010178\\U00010179-\\U00010189\\U0001018A-\\U0001018B\\U0001018C-\\U0001018E\\U000101A0\\U0001D200-\\U0001D241\\U0001D242-\\U0001D244\\U0001D245]
3	Cyrillic	[\\u0400-\\u0481\\u0482\\u0483-\\u0484\\u0487\\u0488-\\u0489\\u048A-\\u052F\\u1C80-\\u1C8A\\u1D2B\\u1D78\\u2DE0-\\u2DFF\\uA640-\\uA66D\\uA66E\\uA66F\\uA670-\\uA672\\uA673\\uA674-\\uA67D\\uA67E\\uA67F\\uA680-\\uA69B\\uA69C-\\uA69D\\uA69E-\\uA69F\\uFE2E-\\uFE2F\\U0001E030-\\U0001E06D\\U0001E08F]
4	Armenian	[\\u0531-\\u0556\\u0559\\u055A-\\u055F\\u0560-\\u0588\\u0589\\u058A\\u058D-\\u058E\\u058F\\uFB13-\\uFB17]
5	Hebrew	[\\u0591-\\u05BD\\u05BE\\u05BF\\u05C0\\u05C1-\\u05C2\\u05C3\\u05C4-\\u05C5\\u05C6\\u05C7\\u05D0-\\u05EA\\u05EF-\\u05F2\\u05F3-\\u05F4\\uFB1D\\uFB1E\\uFB1F-\\uFB28\\uFB29\\uFB2A-\\uFB36\\uFB38-\\uFB3C\\uFB3E\\uFB40-\\uFB41\\uFB43-\\uFB44\\uFB46-\\uFB4F]

 doi:10.20944/preprints202512.1082.v1

6	Arabic	[\\u0600-\\u0604\\u0606-\\u0608\\u0609- \\u060A\\u060B\\u060D\\u060E-\\u060F\\u0610- \\u061A\\u061C\\u061D-\\u061E\\u0620-\\u063F\\u0641- \\u064A\\u0656-\\u065F\\u0660-\\u0669\\u066A-\\u066D\\u066E- \\u066F\\u0671-\\u06D3\\u06D4\\u06D5\\u06D6- \\u06DC\\u06DE\\u06DF-\\u06E4\\u06E5-\\u06E6\\u06E7- \\u06E8\\u06E9\\u06EA-\\u06ED\\u06EE-\\u06EF\\u06F0- \\u06F9\\u06FA-\\u06FC\\u06FD-\\u06FE\\u06FF\\u0750- \\u077F\\u0870-\\u0887\\u0888\\u0889-\\u088E\\u0890- \\u0891\\u0897-\\u089F\\u08A0-\\u08C8\\u08C9\\u08CA- \\u08E1\\u08E3-\\u08FF\\uFB50-\\uFBB1\\uFBB2-\\uFBC2\\uFBD3- \\uFD3D\\uFD40-\\uFD4F\\uFD50-\\uFD8F\\uFD92- \\uFDC7\\uFDCF\\uFDF0-\\uFDFB\\uFDFC\\uFDFD- \\uFDFF\\uFE70-\\uFE74\\uFE76-\\uFEFC\\U00010E60- \\U00010E7E\\U00010EC2-\\U00010EC4\\U00010EFC- \\U00010EFF\\U0001EE00-\\U0001EE03\\U0001EE05- \\U0001EE1F\\U0001EE21- \\U0001EE22\\U0001EE24\\U0001EE27\\U0001EE29- \\U0001EE32\\U0001EE34- \\U0001EE37\\U0001EE39\\U0001EE3B\\U0001EE42\\U0001EE47\\U0001EE49\\U0001EE4B\\U0001EE4D-\\U0001EE4F\\U0001EE51- \\U0001EE52\\U0001EE54\\U0001EE57\\U0001EE59\\U0001EE5B\\U0001EE5D\\U0001EE5F\\U0001EE61-\\U0001EE62\\U0001EE64\\U0001EE67- \\U0001EE6A\\U0001EE6C-\\U0001EE72\\U0001EE74- \\U0001EE77\\U0001EE79-\\U0001EE7C\\U0001EE7E\\U0001EE80- \\U0001EE89\\U0001EE8B-\\U0001EE9B\\U0001EEA1- \\U0001EEA3\\U0001EEA5-\\U0001EEA9\\U0001EEAB- \\U0001EEBB\\U0001EEF0-\\U0001EEF1]
7	Syriac	[\\u0700-\\u070D\\u070F\\u0710\\u0711\\u0712-\\u072F\\u0730- \\u074A\\u074D-\\u074F\\u0860-\\u086A]
8	Thaana	[\\u0780-\\u07A5\\u07A6-\\u07B0\\u07B1]
9	Devanagari	[\\u0900-\\u0902\\u0903\\u0904- \\u0939\\u093A\\u093B\\u093C\\u093D\\u093E-\\u0940\\u0941- \\u0948\\u0949-\\u094C\\u094D\\u094E-\\u094F\\u0950\\u0955- \\u0957\\u0958-\\u0961\\u0962-\\u0963\\u0966- \\u096F\\u0970\\u0971\\u0972-\\u097F\\uA8E0-\\uA8F1\\uA8F2- \\uA8F7\\uA8F8-\\uA8FA\\uA8FB\\uA8FC\\uA8FD- \\uA8FE\\uA8FF\\U00011B00-\\U00011B09]
10	Bengali	[\\u0980\\u0981\\u0982-\\u0983\\u0985-\\u098C\\u098F- \\u0990\\u0993-\\u09A8\\u09AA-\\u09B0\\u09B2\\u09B6- \\u09B9\\u09BC\\u09BD\\u09BE-\\u09C0\\u09C1-\\u09C4\\u09C7- \\u09C8\\u09CB-\\u09CC\\u09CD\\u09CE\\u09D7\\u09DC- \\u09DD\\u09DF-\\u09E1\\u09E2-\\u09E3\\u09E6-\\u09EF\\u09F0- \\u09F1\\u09F2-\\u09F3\\u09F4- \\u09F9\\u09FA\\u09FB\\u09FC\\u09FD\\u09FE]
11	Gurmukhi	[\\u0A01-\\u0A02\\u0A03\\u0A05-\\u0A0A\\u0A0F-\\u0A10\\u0A13- \\u0A28\\u0A2A-\\u0A30\\u0A32-\\u0A33\\u0A35-\\u0A36\\u0A38- \\u0A39\\u0A3C\\u0A3E-\\u0A40\\u0A41-\\u0A42\\u0A47- \\u0A48\\u0A4B-\\u0A4D\\u0A51\\u0A59-\\u0A5C\\u0A5E\\u0A66- \\u0A6F\\u0A70-\\u0A71\\u0A72-\\u0A74\\u0A75\\u0A76]

 doi:10.20944/preprints202512.1082.v1

12	Gujarati	[\\u0A81-\\u0A82\\u0A83\\u0A85-\\u0A8D\\u0A8F-\\u0A91\\u0A93- \\u0AA8\\u0AAA-\\u0AB0\\u0AB2-\\u0AB3\\u0AB5- \\u0AB9\\u0ABC\\u0ABD\\u0ABE-\\u0AC0\\u0AC1- \\u0AC5\\u0AC7-\\u0AC8\\u0AC9\\u0ACB- \\u0ACC\\u0ACD\\u0AD0\\u0AE0-\\u0AE1\\u0AE2- \\u0AE3\\u0AE6-\\u0AEF\\u0AF0\\u0AF1\\u0AF9\\u0AFA-\\u0AFF]
13	Oriya	[\\u0B01\\u0B02-\\u0B03\\u0B05-\\u0B0C\\u0B0F-\\u0B10\\u0B13- \\u0B28\\u0B2A-\\u0B30\\u0B32-\\u0B33\\u0B35- \\u0B39\\u0B3C\\u0B3D\\u0B3E\\u0B3F\\u0B40\\u0B41- \\u0B44\\u0B47-\\u0B48\\u0B4B-\\u0B4C\\u0B4D\\u0B55- \\u0B56\\u0B57\\u0B5C-\\u0B5D\\u0B5F-\\u0B61\\u0B62- \\u0B63\\u0B66-\\u0B6F\\u0B70\\u0B71\\u0B72-\\u0B77]
14	Tamil	[\\u0B82\\u0B83\\u0B85-\\u0B8A\\u0B8E-\\u0B90\\u0B92- \\u0B95\\u0B99-\\u0B9A\\u0B9C\\u0B9E-\\u0B9F\\u0BA3- \\u0BA4\\u0BA8-\\u0BAA\\u0BAE-\\u0BB9\\u0BBE- \\u0BBF\\u0BC0\\u0BC1-\\u0BC2\\u0BC6-\\u0BC8\\u0BCA- \\u0BCC\\u0BCD\\u0BD0\\u0BD7\\u0BE6-\\u0BEF\\u0BF0- \\u0BF2\\u0BF3-\\u0BF8\\u0BF9\\u0BFA\\U00011FC0- \\U00011FD4\\U00011FD5-\\U00011FDC\\U00011FDD- \\U00011FE0\\U00011FE1-\\U00011FF1\\U00011FFF]
15	Telugu	[\\u0C00\\u0C01-\\u0C03\\u0C04\\u0C05-\\u0C0C\\u0C0E- \\u0C10\\u0C12-\\u0C28\\u0C2A-\\u0C39\\u0C3C\\u0C3D\\u0C3E- \\u0C40\\u0C41-\\u0C44\\u0C46-\\u0C48\\u0C4A-\\u0C4D\\u0C55- \\u0C56\\u0C58-\\u0C5A\\u0C5D\\u0C60-\\u0C61\\u0C62- \\u0C63\\u0C66-\\u0C6F\\u0C77\\u0C78-\\u0C7E\\u0C7F]
16	Kannada	[\\u0C80\\u0C81\\u0C82-\\u0C83\\u0C84\\u0C85-\\u0C8C\\u0C8E- \\u0C90\\u0C92-\\u0CA8\\u0CAA-\\u0CB3\\u0CB5- \\u0CB9\\u0CBC\\u0CBD\\u0CBE\\u0CBF\\u0CC0- \\u0CC4\\u0CC6\\u0CC7-\\u0CC8\\u0CCA-\\u0CCB\\u0CCC- \\u0CCD\\u0CD5-\\u0CD6\\u0CDD-\\u0CDE\\u0CE0- \\u0CE1\\u0CE2-\\u0CE3\\u0CE6-\\u0CEF\\u0CF1-\\u0CF2\\u0CF3]
17	Malayalam	[\\u0D00-\\u0D01\\u0D02-\\u0D03\\u0D04-\\u0D0C\\u0D0E- \\u0D10\\u0D12-\\u0D3A\\u0D3B-\\u0D3C\\u0D3D\\u0D3E- \\u0D40\\u0D41-\\u0D44\\u0D46-\\u0D48\\u0D4A- \\u0D4C\\u0D4D\\u0D4E\\u0D4F\\u0D54-\\u0D56\\u0D57\\u0D58- \\u0D5E\\u0D5F-\\u0D61\\u0D62-\\u0D63\\u0D66-\\u0D6F\\u0D70- \\u0D78\\u0D79\\u0D7A-\\u0D7F]
18	Sinhala	[\\u0D81\\u0D82-\\u0D83\\u0D85-\\u0D96\\u0D9A-\\u0DB1\\u0DB3- \\u0DBB\\u0DBD\\u0DC0-\\u0DC6\\u0DCA\\u0DCF- \\u0DD1\\u0DD2-\\u0DD4\\u0DD6\\u0DD8-\\u0DDF\\u0DE6- \\u0DEF\\u0DF2-\\u0DF3\\u0DF4\\U000111E1-\\U000111F4]
19	Thai	[\\u0E01-\\u0E30\\u0E31\\u0E32-\\u0E33\\u0E34-\\u0E3A\\u0E40- \\u0E45\\u0E46\\u0E47-\\u0E4E\\u0E4F\\u0E50-\\u0E59\\u0E5A- \\u0E5B]
20	Lao	[\\u0E81-\\u0E82\\u0E84\\u0E86-\\u0E8A\\u0E8C- \\u0EA3\\u0EA5\\u0EA7-\\u0EB0\\u0EB1\\u0EB2-\\u0EB3\\u0EB4- \\u0EBC\\u0EBD\\u0EC0-\\u0EC4\\u0EC6\\u0EC8-\\u0ECE\\u0ED0- \\u0ED9\\u0EDC-\\u0EDF]

 doi:10.20944/preprints202512.1082.v1

21	Tibetan	[\\u0F00\\u0F01-\\u0F03\\u0F04-\\u0F12\\u0F13\\u0F14\\u0F15-\\u0F17\\u0F18-\\u0F19\\u0F1A-\\u0F1F\\u0F20-\\u0F29\\u0F2A-\\u0F33\\u0F34\\u0F35\\u0F36\\u0F37\\u0F38\\u0F39\\u0F3A\\u0F3B\\u0F3C\\u0F3D\\u0F3E-\\u0F3F\\u0F40-\\u0F47\\u0F49-\\u0F6C\\u0F71-\\u0F7E\\u0F7F\\u0F80-\\u0F84\\u0F85\\u0F86-\\u0F87\\u0F88-\\u0F8C\\u0F8D-\\u0F97\\u0F99-\\u0FBC\\u0FBE-\\u0FC5\\u0FC6\\u0FC7-\\u0FCC\\u0FCE-\\u0FCF\\u0FD0-\\u0FD4\\u0FD9-\\u0FDA]
22	Myanmar	[\\u1000-\\u102A\\u102B-\\u102C\\u102D-\\u1030\\u1031\\u1032-\\u1037\\u1038\\u1039-\\u103A\\u103B-\\u103C\\u103D-\\u103E\\u103F\\u1040-\\u1049\\u104A-\\u104F\\u1050-\\u1055\\u1056-\\u1057\\u1058-\\u1059\\u105A-\\u105D\\u105E-\\u1060\\u1061\\u1062-\\u1064\\u1065-\\u1066\\u1067-\\u106D\\u106E-\\u1070\\u1071-\\u1074\\u1075-\\u1081\\u1082\\u1083-\\u1084\\u1085-\\u1086\\u1087-\\u108C\\u108D\\u108E\\u108F\\u1090-\\u1099\\u109A-\\u109C\\u109D\\u109E-\\u109F\\uA9E0-\\uA9E4\\uA9E5\\uA9E6\\uA9E7-\\uA9EF\\uA9F0-\\uA9F9\\uA9FA-\\uA9FE\\uAA60-\\uAA6F\\uAA70\\uAA71-\\uAA76\\uAA77-\\uAA79\\uAA7A\\uAA7B\\uAA7C\\uAA7D\\uAA7E-\\uAA7F\\U000116D0-\\U000116E3]
23	Georgian	[\\u10A0-\\u10C5\\u10C7\\u10CD\\u10D0-\\u10FA\\u10FC\\u10FD-\\u10FF\\u1C90-\\u1CBA\\u1CBD-\\u1CBF\\u2D00-\\u2D25\\u2D27\\u2D2D]
24	Hangul	[\\u1100-\\u11FF\\u302E-\\u302F\\u3131-\\u318E\\u3200-\\u321E\\u3260-\\u327E\\uA960-\\uA97C\\uAC00-\\uD7A3\\uD7B0-\\uD7C6\\uD7CB-\\uD7FB\\uFFA0-\\uFFBE\\uFFC2-\\uFFC7\\uFFCA-\\uFFCF\\uFFD2-\\uFFD7\\uFFDA-\\uFFDC]
25	Ethiopic	[\\u1200-\\u1248\\u124A-\\u124D\\u1250-\\u1256\\u1258\\u125A-\\u125D\\u1260-\\u1288\\u128A-\\u128D\\u1290-\\u12B0\\u12B2-\\u12B5\\u12B8-\\u12BE\\u12C0\\u12C2-\\u12C5\\u12C8-\\u12D6\\u12D8-\\u1310\\u1312-\\u1315\\u1318-\\u135A\\u135D-\\u135F\\u1360-\\u1368\\u1369-\\u137C\\u1380-\\u138F\\u1390-\\u1399\\u2D80-\\u2D96\\u2DA0-\\u2DA6\\u2DA8-\\u2DAE\\u2DB0-\\u2DB6\\u2DB8-\\u2DBE\\u2DC0-\\u2DC6\\u2DC8-\\u2DCE\\u2DD0-\\u2DD6\\u2DD8-\\u2DDE\\uAB01-\\uAB06\\uAB09-\\uAB0E\\uAB11-\\uAB16\\uAB20-\\uAB26\\uAB28-\\uAB2E\\U0001E7E0-\\U0001E7E6\\U0001E7E8-\\U0001E7EB\\U0001E7ED-\\U0001E7EE\\U0001E7F0-\\U0001E7FE]
26	Cherokee	[\\u13A0-\\u13F5\\u13F8-\\u13FD\\uAB70-\\uABBF]
27	Canadian_ Aboriginal	[\\u1400\\u1401-\\u166C\\u166D\\u166E\\u166F-\\u167F\\u18B0-\\u18F5\\U00011AB0-\\U00011ABF]
28	Ogham	[\\u1680\\u1681-\\u169A\\u169B\\u169C]
29	Runic	[\\u16A0-\\u16EA\\u16EE-\\u16F0\\u16F1-\\u16F8]
30	Khmer	[\\u1780-\\u17B3\\u17B4-\\u17B5\\u17B6\\u17B7-\\u17BD\\u17BE-\\u17C5\\u17C6\\u17C7-\\u17C8\\u17C9-\\u17D3\\u17D4-\\u17D6\\u17D7\\u17D8-\\u17DA\\u17DB\\u17DC\\u17DD\\u17E0-\\u17E9\\u17F0-\\u17F9\\u19E0-\\u19FF]
31	Mongolian	[\\u1800-\\u1801\\u1804\\u1806\\u1807-\\u180A\\u180B-\\u180D\\u180E\\u180F\\u1810-\\u1819\\u1820-\\u1842\\u1843\\u1844-\\u1878\\u1880-\\u1884\\u1885-\\u1886\\u1887-\\u18A8\\u18A9\\u18AA\\U00011660-\\U0001166C]

 doi:10.20944/preprints202512.1082.v1

32	Hiragana	[\\u3041-\\u3096\\u309D-\\u309E\\u309F\\U0001B001- \\U0001B11F\\U0001B132\\U0001B150-\\U0001B152\\U0001F200]
33	Katakana	[\\u30A1-\\u30FA\\u30FD-\\u30FE\\u30FF\\u31F0-\\u31FF\\u32D0- \\u32FE\\u3300-\\u3357\\uFF66-\\uFF6F\\uFF71- \\uFF9D\\U0001AFF0-\\U0001AFF3\\U0001AFF5- \\U0001AFFB\\U0001AFFD-\\U0001AFFE\\U0001B000\\U0001B120- \\U0001B122\\U0001B155\\U0001B164-\\U0001B167]
34	Bopomofo	[\\u02EA-\\u02EB\\u3105-\\u312F\\u31A0-\\u31BF]
35	Han	[\\u2E80-\\u2E99\\u2E9B-\\u2EF3\\u2F00- \\u2FD5\\u3005\\u3007\\u3021-\\u3029\\u3038- \\u303A\\u303B\\u3400-\\u4DBF\\u4E00-\\u9FFF\\uF900- \\uFA6D\\uFA70-\\uFAD9\\U00016FE2\\U00016FE3\\U00016FF0- \\U00016FF1\\U00020000-\\U0002A6DF\\U0002A700- \\U0002B739\\U0002B740-\\U0002B81D\\U0002B820- \\U0002CEA1\\U0002CEB0-\\U0002EBE0\\U0002EBF0- \\U0002EE5D\\U0002F800-\\U0002FA1D\\U00030000- \\U0003134A\\U00031350-\\U000323AF]
36	Yi	[\\uA000-\\uA014\\uA015\\uA016-\\uA48C\\uA490-\\uA4C6]
37	Old_Italic	[\\U00010300-\\U0001031F\\U00010320-\\U00010323\\U0001032D- \\U0001032F]
38	Gothic	[\\U00010330-\\U00010340\\U00010341\\U00010342- \\U00010349\\U0001034A]
39	Deseret	[\\U00010400-\\U0001044F]
40	Inherited	[\\u0300-\\u036F\\u0485-\\u0486\\u064B-\\u0655\\u0670\\u0951- \\u0954\\u1AB0-\\u1ABD\\u1ABE\\u1ABF-\\u1ACE\\u1CD0- \\u1CD2\\u1CD4-\\u1CE0\\u1CE2-\\u1CE8\\u1CED\\u1CF4\\u1CF8- \\u1CF9\\u1DC0-\\u1DFF\\u200C-\\u200D\\u20D0-\\u20DC\\u20DD- \\u20E0\\u20E1\\u20E2-\\u20E4\\u20E5-\\u20F0\\u302A- \\u302D\\u3099-\\u309A\\uFE00-\\uFE0F\\uFE20- \\uFE2D\\U000101FD\\U000102E0\\U0001133B\\U0001CF00- \\U0001CF2D\\U0001CF30-\\U0001CF46\\U0001D167- \\U0001D169\\U0001D17B-\\U0001D182\\U0001D185- \\U0001D18B\\U0001D1AA-\\U0001D1AD\\U000E0100-\\U000E01EF]
41	Tagalog	[\\u1700-\\u1711\\u1712-\\u1714\\u1715\\u171F]
42	Hanunoo	[\\u1720-\\u1731\\u1732-\\u1733\\u1734]
43	Buhid	[\\u1740-\\u1751\\u1752-\\u1753]
44	Tagbanwa	[\\u1760-\\u176C\\u176E-\\u1770\\u1772-\\u1773]
45	Limbu	[\\u1900-\\u191E\\u1920-\\u1922\\u1923-\\u1926\\u1927- \\u1928\\u1929-\\u192B\\u1930-\\u1931\\u1932\\u1933- \\u1938\\u1939-\\u193B\\u1940\\u1944-\\u1945\\u1946-\\u194F]
46	Tai_Le	[\\u1950-\\u196D\\u1970-\\u1974]
47	Linear_B	[\\U00010000-\\U0001000B\\U0001000D-\\U00010026\\U00010028- \\U0001003A\\U0001003C-\\U0001003D\\U0001003F- \\U0001004D\\U00010050-\\U0001005D\\U00010080-\\U000100FA]
48	Ugaritic	[\\U00010380-\\U0001039D\\U0001039F]
49	Shavian	[\\U00010450-\\U0001047F]
50	Osmanya	[\\U00010480-\\U0001049D\\U000104A0-\\U000104A9]
51	Cypriot	[\\U00010800-\\U00010805\\U00010808\\U0001080A- \\U00010835\\U00010837-\\U00010838\\U0001083C\\U0001083F]
52	Braille	[\\u2800-\\u28FF]
53	Buginese	[\\u1A00-\\u1A16\\u1A17-\\u1A18\\u1A19-\\u1A1A\\u1A1B\\u1A1E- \\u1A1F]

 doi:10.20944/preprints202512.1082.v1

54	Coptic	[\\u03E2-\\u03EF\\u2C80-\\u2CE4\\u2CE5-\\u2CEA\\u2CEB-\\u2CEE\\u2CEF-\\u2CF1\\u2CF2-\\u2CF3\\u2CF9-\\u2CFC\\u2CFD\\u2CFE-\\u2CFF]
55	New_Tai_Lue	[\\u1980-\\u19AB\\u19B0-\\u19C9\\u19D0-\\u19D9\\u19DA\\u19DE-\\u19DF]
56	Glagolitic	[\\u2C00-\\u2C5F\\U0001E000-\\U0001E006\\U0001E008-\\U0001E018\\U0001E01B-\\U0001E021\\U0001E023-\\U0001E024\\U0001E026-\\U0001E02A]
57	Tifinagh	[\\u2D30-\\u2D67\\u2D6F\\u2D70\\u2D7F]
58	Syloti_Nagri	[\\uA800-\\uA801\\uA802\\uA803-\\uA805\\uA806\\uA807-\\uA80A\\uA80B\\uA80C-\\uA822\\uA823-\\uA824\\uA825-\\uA826\\uA827\\uA828-\\uA82B\\uA82C]
59	Old_Persian	[\\U000103A0-\\U000103C3\\U000103C8-\\U000103CF\\U000103D0\\U000103D1-\\U000103D5]
60	Kharoshthi	[\\U00010A00\\U00010A01-\\U00010A03\\U00010A05-\\U00010A06\\U00010A0C-\\U00010A0F\\U00010A10-\\U00010A13\\U00010A15-\\U00010A17\\U00010A19-\\U00010A35\\U00010A38-\\U00010A3A\\U00010A3F\\U00010A40-\\U00010A48\\U00010A50-\\U00010A58]
61	Balinese	[\\u1B00-\\u1B03\\u1B04\\u1B05-\\u1B33\\u1B34\\u1B35\\u1B36-\\u1B3A\\u1B3B\\u1B3C\\u1B3D-\\u1B41\\u1B42\\u1B43-\\u1B44\\u1B45-\\u1B4C\\u1B4E-\\u1B4F\\u1B50-\\u1B59\\u1B5A-\\u1B60\\u1B61-\\u1B6A\\u1B6B-\\u1B73\\u1B74-\\u1B7C\\u1B7D-\\u1B7F]
62	Cuneiform	[\\U00012000-\\U00012399\\U00012400-\\U0001246E\\U00012470-\\U00012474\\U00012480-\\U00012543]
63	Phoenician	[\\U00010900-\\U00010915\\U00010916-\\U0001091B\\U0001091F]
64	Phags_Pa	[\\uA840-\\uA873\\uA874-\\uA877]
65	Nko	[\\u07C0-\\u07C9\\u07CA-\\u07EA\\u07EB-\\u07F3\\u07F4-\\u07F5\\u07F6\\u07F7-\\u07F9\\u07FA\\u07FD\\u07FE-\\u07FF]
66	Sundanese	[\\u1B80-\\u1B81\\u1B82\\u1B83-\\u1BA0\\u1BA1\\u1BA2-\\u1BA5\\u1BA6-\\u1BA7\\u1BA8-\\u1BA9\\u1BAA\\u1BAB-\\u1BAD\\u1BAE-\\u1BAF\\u1BB0-\\u1BB9\\u1BBA-\\u1BBF\\u1CC0-\\u1CC7]
67	Lepcha	[\\u1C00-\\u1C23\\u1C24-\\u1C2B\\u1C2C-\\u1C33\\u1C34-\\u1C35\\u1C36-\\u1C37\\u1C3B-\\u1C3F\\u1C40-\\u1C49\\u1C4D-\\u1C4F]
68	Ol_Chiki	[\\u1C50-\\u1C59\\u1C5A-\\u1C77\\u1C78-\\u1C7D\\u1C7E-\\u1C7F]
69	Vai	[\\uA500-\\uA60B\\uA60C\\uA60D-\\uA60F\\uA610-\\uA61F\\uA620-\\uA629\\uA62A-\\uA62B]
70	Saurashtra	[\\uA880-\\uA881\\uA882-\\uA8B3\\uA8B4-\\uA8C3\\uA8C4-\\uA8C5\\uA8CE-\\uA8CF\\uA8D0-\\uA8D9]
71	Kayah_Li	[\\uA900-\\uA909\\uA90A-\\uA925\\uA926-\\uA92D\\uA92F]
72	Rejang	[\\uA930-\\uA946\\uA947-\\uA951\\uA952-\\uA953\\uA95F]
73	Lycian	[\\U00010280-\\U0001029C]
74	Carian	[\\U000102A0-\\U000102D0]
75	Lydian	[\\U00010920-\\U00010939\\U0001093F]
76	Cham	[\\uAA00-\\uAA28\\uAA29-\\uAA2E\\uAA2F-\\uAA30\\uAA31-\\uAA32\\uAA33-\\uAA34\\uAA35-\\uAA36\\uAA40-\\uAA42\\uAA43\\uAA44-\\uAA4B\\uAA4C\\uAA4D\\uAA50-\\uAA59\\uAA5C-\\uAA5F]

 doi:10.20944/preprints202512.1082.v1

77	Tai_Tham	[\\u1A20-\\u1A54\\u1A55\\u1A56\\u1A57\\u1A58-\\u1A5E\\u1A60\\u1A61\\u1A62\\u1A63-\\u1A64\\u1A65-\\u1A6C\\u1A6D-\\u1A72\\u1A73-\\u1A7C\\u1A7F\\u1A80-\\u1A89\\u1A90-\\u1A99\\u1AA0-\\u1AA6\\u1AA7\\u1AA8-\\u1AAD]
78	Tai_Viet	[\\uAA80-\\uAAAF\\uAAB0\\uAAB1\\uAAB2-\\uAAB4\\uAAB5-\\uAAB6\\uAAB7-\\uAAB8\\uAAB9-\\uAABD\\uAABE-\\uAABF\\uAAC0\\uAAC1\\uAAC2\\uAADB-\\uAADC\\uAADD\\uAADE-\\uAADF]
79	Avestan	[\\U00010B00-\\U00010B35\\U00010B39-\\U00010B3F]
80	Egyptian_Hieroglyphs	[\\U00013000-\\U0001342F\\U00013430-\\U0001343F\\U00013440\\U00013441-\\U00013446\\U00013447-\\U00013455\\U00013460-\\U000143FA]
81	Samaritan	[\\u0800-\\u0815\\u0816-\\u0819\\u081A\\u081B-\\u0823\\u0824\\u0825-\\u0827\\u0828\\u0829-\\u082D\\u0830-\\u083E]
82	Lisu	[\\uA4D0-\\uA4F7\\uA4F8-\\uA4FD\\uA4FE-\\uA4FF\\U00011FB0]
83	Bamum	[\\uA6A0-\\uA6E5\\uA6E6-\\uA6EF\\uA6F0-\\uA6F1\\uA6F2-\\uA6F7\\U00016800-\\U00016A38]
84	Javanese	[\\uA980-\\uA982\\uA983\\uA984-\\uA9B2\\uA9B3\\uA9B4-\\uA9B5\\uA9B6-\\uA9B9\\uA9BA-\\uA9BB\\uA9BC-\\uA9BD\\uA9BE-\\uA9C0\\uA9C1-\\uA9CD\\uA9D0-\\uA9D9\\uA9DE-\\uA9DF]
85	Meetei_Mayek	[\\uAAE0-\\uAAEA\\uAAEB\\uAAEC-\\uAAED\\uAAEE-\\uAAEF\\uAAF0-\\uAAF1\\uAAF2\\uAAF3-\\uAAF4\\uAAF5\\uAAF6\\uABC0-\\uABE2\\uABE3-\\uABE4\\uABE5\\uABE6-\\uABE7\\uABE8\\uABE9-\\uABEA\\uABEB\\uABEC\\uABED\\uABF0-\\uABF9]
86	Imperial_Aramaic	[\\U00010840-\\U00010855\\U00010857\\U00010858-\\U0001085F]
87	Old_South_Arabian	[\\U00010A60-\\U00010A7C\\U00010A7D-\\U00010A7E\\U00010A7F]
88	Inscriptional_Parthian	[\\U00010B40-\\U00010B55\\U00010B58-\\U00010B5F]
89	Inscriptional_Pahlavi	[\\U00010B60-\\U00010B72\\U00010B78-\\U00010B7F]
90	Old_Turkic	[\\U00010C00-\\U00010C48]
91	Kaithi	[\\U00011080-\\U00011081\\U00011082\\U00011083-\\U000110AF\\U000110B0-\\U000110B2\\U000110B3-\\U000110B6\\U000110B7-\\U000110B8\\U000110B9-\\U000110BA\\U000110BB-\\U000110BC\\U000110BD\\U000110BE-\\U000110C1\\U000110C2\\U000110CD]
92	Batak	[\\u1BC0-\\u1BE5\\u1BE6\\u1BE7\\u1BE8-\\u1BE9\\u1BEA-\\u1BEC\\u1BED\\u1BEE\\u1BEF-\\u1BF1\\u1BF2-\\u1BF3\\u1BFC-\\u1BFF]
93	Brahmi	[\\U00011000\\U00011001\\U00011002\\U00011003-\\U00011037\\U00011038-\\U00011046\\U00011047-\\U0001104D\\U00011052-\\U00011065\\U00011066-\\U0001106F\\U00011070\\U00011071-\\U00011072\\U00011073-\\U00011074\\U00011075\\U0001107F]
94	Mandaic	[\\u0840-\\u0858\\u0859-\\u085B\\u085E]

 doi:10.20944/preprints202512.1082.v1

95	Chakma	[\\U00011100-\\U00011102\\U00011103-\\U00011126\\U00011127-\\U0001112B\\U0001112C\\U0001112D-\\U00011134\\U00011136-\\U0001113F\\U00011140-\\U00011143\\U00011144\\U00011145-\\U00011146\\U00011147]
96	Meroitic_Cursive	[\\U000109A0-\\U000109B7\\U000109BC-\\U000109BD\\U000109BE-\\U000109BF\\U000109C0-\\U000109CF\\U000109D2-\\U000109FF]
97	Meroitic_Hieroglyphs	[\\U00010980-\\U0001099F]
98	Miao	[\\U00016F00-\\U00016F4A\\U00016F4F\\U00016F50\\U00016F51-\\U00016F87\\U00016F8F-\\U00016F92\\U00016F93-\\U00016F9F]
99	Sharada	[\\U00011180-\\U00011181\\U00011182\\U00011183-\\U000111B2\\U000111B3-\\U000111B5\\U000111B6-\\U000111BE\\U000111BF-\\U000111C0\\U000111C1-\\U000111C4\\U000111C5-\\U000111C8\\U000111C9-\\U000111CC\\U000111CD\\U000111CE\\U000111CF\\U000111D0-\\U000111D9\\U000111DA\\U000111DB\\U000111DC\\U000111DD-\\U000111DF]
100	Sora_Sompeng	[\\U000110D0-\\U000110E8\\U000110F0-\\U000110F9]
101	Takri	[\\U00011680-\\U000116AA\\U000116AB\\U000116AC\\U000116AD\\U000116AE-\\U000116AF\\U000116B0-\\U000116B5\\U000116B6\\U000116B7\\U000116B8\\U000116B9\\U000116C0-\\U000116C9]
102	Caucasian_Albanian	[\\U00010530-\\U00010563\\U0001056F]
103	Bassa_Vah	[\\U00016AD0-\\U00016AED\\U00016AF0-\\U00016AF4\\U00016AF5]
104	Duployan	[\\U0001BC00-\\U0001BC6A\\U0001BC70-\\U0001BC7C\\U0001BC80-\\U0001BC88\\U0001BC90-\\U0001BC99\\U0001BC9C\\U0001BC9D-\\U0001BC9E\\U0001BC9F]
105	Elbasan	[\\U00010500-\\U00010527]
106	Grantha	[\\U00011300-\\U00011301\\U00011302-\\U00011303\\U00011305-\\U0001130C\\U0001130F-\\U00011310\\U00011313-\\U00011328\\U0001132A-\\U00011330\\U00011332-\\U00011333\\U00011335-\\U00011339\\U0001133C\\U0001133D\\U0001133E-\\U0001133F\\U00011340\\U00011341-\\U00011344\\U00011347-\\U00011348\\U0001134B-\\U0001134D\\U00011350\\U00011357\\U0001135D-\\U00011361\\U00011362-\\U00011363\\U00011366-\\U0001136C\\U00011370-\\U00011374]
107	Pahawh_Hmong	[\\U00016B00-\\U00016B2F\\U00016B30-\\U00016B36\\U00016B37-\\U00016B3B\\U00016B3C-\\U00016B3F\\U00016B40-\\U00016B43\\U00016B44\\U00016B45\\U00016B50-\\U00016B59\\U00016B5B-\\U00016B61\\U00016B63-\\U00016B77\\U00016B7D-\\U00016B8F]
108	Khojki	[\\U00011200-\\U00011211\\U00011213-\\U0001122B\\U0001122C-\\U0001122E\\U0001122F-\\U00011231\\U00011232-\\U00011233\\U00011234\\U00011235\\U00011236-\\U00011237\\U00011238-\\U0001123D\\U0001123E\\U0001123F-\\U00011240\\U00011241]
109	Linear_A	[\\U00010600-\\U00010736\\U00010740-\\U00010755\\U00010760-\\U00010767]

 doi:10.20944/preprints202512.1082.v1

110	Mahajani	[\\U00011150-\\U00011172\\U00011173\\U00011174-\\U00011175\\U00011176]
111	Manichaean	[\\U00010AC0-\\U00010AC7\\U00010AC8\\U00010AC9-\\U00010AE4\\U00010AE5-\\U00010AE6\\U00010AEB-\\U00010AEF\\U00010AF0-\\U00010AF6]
112	Mende_Kikakui	[\\U0001E800-\\U0001E8C4\\U0001E8C7-\\U0001E8CF\\U0001E8D0-\\U0001E8D6]
113	Modi	[\\U00011600-\\U0001162F\\U00011630-\\U00011632\\U00011633-\\U0001163A\\U0001163B-\\U0001163C\\U0001163D\\U0001163E\\U0001163F-\\U00011640\\U00011641-\\U00011643\\U00011644\\U00011650-\\U00011659]
114	Mro	[\\U00016A40-\\U00016A5E\\U00016A60-\\U00016A69\\U00016A6E-\\U00016A6F]
115	Old_North_Arabian	[\\U00010A80-\\U00010A9C\\U00010A9D-\\U00010A9F]
116	Nabataean	[\\U00010880-\\U0001089E\\U000108A7-\\U000108AF]
117	Palmyrene	[\\U00010860-\\U00010876\\U00010877-\\U00010878\\U00010879-\\U0001087F]
118	Pau_Cin_Hau	[\\U00011AC0-\\U00011AF8]
119	Old_Permic	[\\U00010350-\\U00010375\\U00010376-\\U0001037A]
120	Psalter_Pahlavi	[\\U00010B80-\\U00010B91\\U00010B99-\\U00010B9C\\U00010BA9-\\U00010BAF]
121	Siddham	[\\U00011580-\\U000115AE\\U000115AF-\\U000115B1\\U000115B2-\\U000115B5\\U000115B8-\\U000115BB\\U000115BC-\\U000115BD\\U000115BE\\U000115BF-\\U000115C0\\U000115C1-\\U000115D7\\U000115D8-\\U000115DB\\U000115DC-\\U000115DD]
122	Khudawadi	[\\U000112B0-\\U000112DE\\U000112DF\\U000112E0-\\U000112E2\\U000112E3-\\U000112EA\\U000112F0-\\U000112F9]
123	Tirhuta	[\\U00011480-\\U000114AF\\U000114B0-\\U000114B2\\U000114B3-\\U000114B8\\U000114B9\\U000114BA\\U000114BB-\\U000114BE\\U000114BF-\\U000114C0\\U000114C1\\U000114C2-\\U000114C3\\U000114C4-\\U000114C5\\U000114C6\\U000114C7\\U000114D0-\\U000114D9]
124	Warang_Citi	[\\U000118A0-\\U000118DF\\U000118E0-\\U000118E9\\U000118EA-\\U000118F2\\U000118FF]
125	Ahom	[\\U00011700-\\U0001171A\\U0001171D\\U0001171E\\U0001171F\\U00011720-\\U00011721\\U00011722-\\U00011725\\U00011726\\U00011727-\\U0001172B\\U00011730-\\U00011739\\U0001173A-\\U0001173B\\U0001173C-\\U0001173E\\U0001173F\\U00011740-\\U00011746]
126	Anatolian_Hieroglyphs	[\\U00014400-\\U00014646]
127	Hatran	[\\U000108E0-\\U000108F2\\U000108F4-\\U000108F5\\U000108FB-\\U000108FF]
128	Multani	[\\U00011280-\\U00011286\\U00011288\\U0001128A-\\U0001128D\\U0001128F-\\U0001129D\\U0001129F-\\U000112A8\\U000112A9]

 doi:10.20944/preprints202512.1082.v1

129	Old_Hungarian	[\\U00010C80-\\U00010CB2\\U00010CC0-\\U00010CF2\\U00010CFA-\\U00010CFF]
130	SignWriting	[\\U0001D800-\\U0001D9FF\\U0001DA00-\\U0001DA36\\U0001DA37-\\U0001DA3A\\U0001DA3B-\\U0001DA6C\\U0001DA6D-\\U0001DA74\\U0001DA75\\U0001DA76-\\U0001DA83\\U0001DA84\\U0001DA85-\\U0001DA86\\U0001DA87-\\U0001DA8B\\U0001DA9B-\\U0001DA9F\\U0001DAA1-\\U0001DAAF]
131	Adlam	[\\U0001E900-\\U0001E943\\U0001E944-\\U0001E94A\\U0001E94B\\U0001E950-\\U0001E959\\U0001E95E-\\U0001E95F]
132	Bhaiksuki	[\\U00011C00-\\U00011C08\\U00011C0A-\\U00011C2E\\U00011C2F\\U00011C30-\\U00011C36\\U00011C38-\\U00011C3D\\U00011C3E\\U00011C3F\\U00011C40\\U00011C41-\\U00011C45\\U00011C50-\\U00011C59\\U00011C5A-\\U00011C6C]
133	Marchen	[\\U00011C70-\\U00011C71\\U00011C72-\\U00011C8F\\U00011C92-\\U00011CA7\\U00011CA9\\U00011CAA-\\U00011CB0\\U00011CB1\\U00011CB2-\\U00011CB3\\U00011CB4\\U00011CB5-\\U00011CB6]
134	Newa	[\\U00011400-\\U00011434\\U00011435-\\U00011437\\U00011438-\\U0001143F\\U00011440-\\U00011441\\U00011442-\\U00011444\\U00011445\\U00011446\\U00011447-\\U0001144A\\U0001144B-\\U0001144F\\U00011450-\\U00011459\\U0001145A-\\U0001145B\\U0001145D\\U0001145E\\U0001145F-\\U00011461]
135	Osage	[\\U000104B0-\\U000104D3\\U000104D8-\\U000104FB]
136	Tangut	[\\U00016FE0\\U00017000-\\U000187F7\\U00018800-\\U00018AFF\\U00018D00-\\U00018D08]
137	Masaram_Gondi	[\\U00011D00-\\U00011D06\\U00011D08-\\U00011D09\\U00011D0B-\\U00011D30\\U00011D31-\\U00011D36\\U00011D3A\\U00011D3C-\\U00011D3D\\U00011D3F-\\U00011D45\\U00011D46\\U00011D47\\U00011D50-\\U00011D59]
138	Nushu	[\\U00016FE1\\U0001B170-\\U0001B2FB]
139	Soyombo	[\\U00011A50\\U00011A51-\\U00011A56\\U00011A57-\\U00011A58\\U00011A59-\\U00011A5B\\U00011A5C-\\U00011A89\\U00011A8A-\\U00011A96\\U00011A97\\U00011A98-\\U00011A99\\U00011A9A-\\U00011A9C\\U00011A9D\\U00011A9E-\\U00011AA2]
140	Zanabazar_Square	[\\U00011A00\\U00011A01-\\U00011A0A\\U00011A0B-\\U00011A32\\U00011A33-\\U00011A38\\U00011A39\\U00011A3A\\U00011A3B-\\U00011A3E\\U00011A3F-\\U00011A46\\U00011A47]
141	Dogra	[\\U00011800-\\U0001182B\\U0001182C-\\U0001182E\\U0001182F-\\U00011837\\U00011838\\U00011839-\\U0001183A\\U0001183B]
142	Gunjala_Gondi	[\\U00011D60-\\U00011D65\\U00011D67-\\U00011D68\\U00011D6A-\\U00011D89\\U00011D8A-\\U00011D8E\\U00011D90-\\U00011D91\\U00011D93-\\U00011D94\\U00011D95\\U00011D96\\U00011D97\\U00011D98\\U00011DA0-\\U00011DA9]
143	Makasar	[\\U00011EE0-\\U00011EF2\\U00011EF3-\\U00011EF4\\U00011EF5-\\U00011EF6\\U00011EF7-\\U00011EF8]
144	Medefaidrin	[\\U00016E40-\\U00016E7F\\U00016E80-\\U00016E96\\U00016E97-\\U00016E9A]

 doi:10.20944/preprints202512.1082.v1

145	Hanifi_Rohingya	[\\U00010D00-\\U00010D23\\U00010D24-\\U00010D27\\U00010D30-\\U00010D39]
146	Sogdian	[\\U00010F30-\\U00010F45\\U00010F46-\\U00010F50\\U00010F51-\\U00010F54\\U00010F55-\\U00010F59]
147	Old_Sogdian	[\\U00010F00-\\U00010F1C\\U00010F1D-\\U00010F26\\U00010F27]
148	Elymaic	[\\U00010FE0-\\U00010FF6]
149	Nandinagari	[\\U000119A0-\\U000119A7\\U000119AA-\\U000119D0\\U000119D1-\\U000119D3\\U000119D4-\\U000119D7\\U000119DA-\\U000119DB\\U000119DC-\\U000119DF\\U000119E0\\U000119E1\\U000119E2\\U000119E3\\U000119E4]
150	Nyiakeng_Puachue_Hmong	[\\U0001E100-\\U0001E12C\\U0001E130-\\U0001E136\\U0001E137-\\U0001E13D\\U0001E140-\\U0001E149\\U0001E14E\\U0001E14F]
151	Wancho	[\\U0001E2C0-\\U0001E2EB\\U0001E2EC-\\U0001E2EF\\U0001E2F0-\\U0001E2F9\\U0001E2FF]
152	Chorasmian	[\\U00010FB0-\\U00010FC4\\U00010FC5-\\U00010FCB]
153	Dives_Akuru	[\\U00011900-\\U00011906\\U00011909\\U0001190C-\\U00011913\\U00011915-\\U00011916\\U00011918-\\U0001192F\\U00011930-\\U00011935\\U00011937-\\U00011938\\U0001193B-\\U0001193C\\U0001193D\\U0001193E\\U0001193F\\U00011940\\U00011941\\U00011942\\U00011943\\U00011944-\\U00011946\\U00011950-\\U00011959]
154	Khitan_Small_Script	[\\U00016FE4\\U00018B00-\\U00018CD5\\U00018CFF]
155	Yezidi	[\\U00010E80-\\U00010EA9\\U00010EAB-\\U00010EAC\\U00010EAD\\U00010EB0-\\U00010EB1]
156	Cypro_Minoan	[\\U00012F90-\\U00012FF0\\U00012FF1-\\U00012FF2]
157	Old_Uyghur	[\\U00010F70-\\U00010F81\\U00010F82-\\U00010F85\\U00010F86-\\U00010F89]
158	Tangsa	[\\U00016A70-\\U00016ABE\\U00016AC0-\\U00016AC9]
159	Toto	[\\U0001E290-\\U0001E2AD\\U0001E2AE]
160	Vithkuqi	[\\U00010570-\\U0001057A\\U0001057C-\\U0001058A\\U0001058C-\\U00010592\\U00010594-\\U00010595\\U00010597-\\U000105A1\\U000105A3-\\U000105B1\\U000105B3-\\U000105B9\\U000105BB-\\U000105BC]
161	Kawi	[\\U00011F00-\\U00011F01\\U00011F02\\U00011F03\\U00011F04-\\U00011F10\\U00011F12-\\U00011F33\\U00011F34-\\U00011F35\\U00011F36-\\U00011F3A\\U00011F3E-\\U00011F3F\\U00011F40\\U00011F41\\U00011F42\\U00011F43-\\U00011F4F\\U00011F50-\\U00011F59\\U00011F5A]
162	Nag_Mundari	[\\U0001E4D0-\\U0001E4EA\\U0001E4EB\\U0001E4EC-\\U0001E4EF\\U0001E4F0-\\U0001E4F9]
163	Garay	[\\U00010D40-\\U00010D49\\U00010D4A-\\U00010D4D\\U00010D4E\\U00010D4F\\U00010D50-\\U00010D65\\U00010D69-\\U00010D6D\\U00010D6E\\U00010D6F\\U00010D70-\\U00010D85\\U00010D8E-\\U00010D8F]

 doi:10.20944/preprints202512.1082.v1

164	Gurung_Khema	[\\U00016100-\\U0001611D\\U0001611E-\\U00016129\\U0001612A-\\U0001612C\\U0001612D-\\U0001612F\\U00016130-\\U00016139]
165	Kirat_Rai	[\\U00016D40-\\U00016D42\\U00016D43-\\U00016D6A\\U00016D6B-\\U00016D6C\\U00016D6D-\\U00016D6F\\U00016D70-\\U00016D79]
166	Ol_Onal	[\\U0001E5D0-\\U0001E5ED\\U0001E5EE-\\U0001E5EF\\U0001E5F0\\U0001E5F1-\\U0001E5FA\\U0001E5FF]
167	Sunuwar	[\\U00011BC0-\\U00011BE0\\U00011BE1\\U00011BF0-\\U00011BF9]
168	Todhri	[\\U000105C0-\\U000105F3]
169	Tulu_Tigalari	[\\U00011380-\\U00011389\\U0001138B\\U0001138E\\U00011390-\\U000113B5\\U000113B7\\U000113B8-\\U000113BA\\U000113BB-\\U000113C0\\U000113C2\\U000113C5\\U000113C7-\\U000113CA\\U000113CC-\\U000113CD\\U000113CE\\U000113CF\\U000113D0\\U000113D1\\U000113D2\\U000113D3\\U000113D4-\\U000113D5\\U000113D7-\\U000113D8\\U000113E1-\\U000113E2]

References

1. Choong, C.Y.; Mikami, Y.; Marasinghe, C.A.; Nandasara, S.T. Optimizing n-gram order of an n-gram based language identification algorithm for 68 written languages. *Int. J. Adv. ICT Emerg. Reg.* **2009**, *2*, 21–28.
2. Toftrup, M.; Srensen, S.A.; Ciosici, M.R.; Assent, I. A reproduction of apple's bi-directional lstm models for language identification in short strings. In Proceedings of the 16th Conference of the European Chapter of the Associationfor Computational Linguistics: Student Research Workshop, Virtual, 19–23 April 2021; pp. 36–42.
3. Botha, G.R.; Barnard, E. Factors that affect the accuracy of text-based language identification. *Comput. Speech Lang.* **2012**, *26*, 307–320.
4. Abainia, K.; Ouamour, S.; Sayoud, H. Effective language identification of forum texts based on statistical approaches. *Inf. Process. Manag. Int. J.* **2016**, *52*, 491–512.
5. Selamat, A.; Akosu, N. Word-length algorithm for language identification of under-resourced languages. *J. King Saud Univ. Comput. Inf. Sci.* **2015**, *28*, 457–469.
6. Jauhiainen, T.; Lui, M.; Zampieri, M.; Baldwin, T.; Linden, K. Automatic language identification in texts: A survey. *J. Artif. Intell. Res.* **2019**, *65*, 675–782.
7. Apple. Language Identification from Very Short Strings. 2019. Available online: https://machinelearning.apple.com/research/language-identification-from-very-short-strings (accessed on 10 February 2021).
8. Maimaitiyiming, H.; Wushour, S. On hierarchical text language-identification algorithms. *Algorithms* **2018**, *11*, 39.
9. Hasimu, M.; Silamu, W. Three-stage short text language identification algorithm. *J. Digit. Inf. Manag.* **2017**, *15*, 354–371.
10. Scripts-16.0.0.txt. Available online: https://www.unicode.org/Public/UNIDATA/Scripts.txt (accessed on 24 January 2025).
11. Hanif, F.; Latif, F.; Khiyal, M.S.H. Unicode Aided Language Identification across Multiple Scripts and Heterogeneous Data. *Inf. Technol. J.* **2007**, *6*, 534–540.
12. Mamtimin, Q.; Wushour, S.; Minghui, Q. The Design of a Script Identification Algorithm and Its Application in Constructing a Text Language Identification Dataset. *Data* **2024**, *9*, 134.
13. Mamtimin, Q.; Wushour, S. Improved Script Identification Algorithm Using Unicode-Based Regular Expression Matching Strategy. Data 2025, 10, 43.
14. Most common languages on the internet. Available online: https://www.statista.com/statistics/262946/most-common-languages-on-the-internet (accessed on 2 December 2025).
15. Scripts-16.0.0.txt. Available online: https://www.unicode.org/Public/UNIDATA/Scripts.txt (accessed on 24 January 2025).

 doi:10.20944/preprints202512.1082.v1

16. Goldhahn, D.T.; Eckart, U. Quasthoff: Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. In Proceedings of the 8th International Language Resources and Evaluation (LREC'12), Istanbul, Turkey, 23–25 May 2012.
17. Leipzig Corpora Collection. Available online: https://cls.corpora.uni-leipzig.de/en (accessed on 28 June 2024).
18. Leipzig Corpora Collection Download Page. Available online: https://wortschatz-leipzig.de/en/download (accessed on 5 July 2024).
19. ISO 639-2 Code. Available online: https://www.loc.gov/standards/iso639-2/php/code_list.php (accessed on 5 July 2024).
20. ISO 639-3 Code. Available online: https://iso639-3.sil.org/code_tables/639/data (accessed on 5 July 2024).