Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 6450–6460
Marseille, 20-25 June 2022

Extensions to Brahmic script processing within the Nisaba library: new scripts, languages and utilities

Alexander Gutkin[†], Cibu Johny[†], Raiomond Doctor[‡*], Lawrence Wolf-Sonkin[◦], Brian Roark[◦]
Google Research
[†]United Kingdom [‡]India [◦]United States
{agutkin,cibu,raiomond,wolfsonkin,roark}@google.com

Abstract

The Brahmic family of scripts is used to record some of the most spoken languages in the world and is arguably the most diverse family of writing systems. In this work, we present several substantial extensions to Brahmic script functionality within the open-source Nisaba library of finite-state script normalization and processing utilities (Johny et al., 2021). First, we extend coverage from the original ten scripts to an additional ten scripts of South Asia and beyond, including some used to record endangered languages such as Dogri. Second, we augment the language layer so that scripts used by multiple languages in distinct ways can be processed correctly for more languages, such as the Bengali script when used for the low-resource language Santali. We document key changes to the finite-state engine required to support these new languages and scripts. Finally, we add new script processing utilities, including lightweight script-level *reading* normalization that (unlike existing visual normalization) does not preserve visual invariance, and a *fixed-input* transliteration mechanism specifically tailored to Brahmic text entry with ASCII characters.

Keywords: finite-state, Brahmic writing systems, low resource, transliteration, normalization

1. Introduction

Nisaba is an open-source library[1] for processing the Brāhmī-derived scripts of South Asia (Johny et al., 2021). Nisaba uses finite-state algorithms that leverage the fundamental organizing principle of such scripts — an orthographic syllable, or akṣara (Coulmas, 1999; Bright, 1999) — to provide a range of script-specific utilities. The design was inspired by the formal algebraic approach to Brahmic scripts advocated by Datta (1984) and substantially advanced by Sproat (2003), and employs finite-state automata (FSAs) and transducers (FSTs) to efficiently and compactly represent several types of finite-state grammars for script normalization, transliteration and validation (Mohri et al., 2000; Gorman, 2016). In this paper, we present several recent important extensions to the library: the addition of many new scripts, some of which were sufficiently distinct to require changes to akṣara modeling; expansion of language (in addition to script) layer functionality, to permit language-specific processing when required; and new normalization and transliteration utilities.

Nisaba originally covered ten major Brahmic scripts of South Asia: Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Oriya (Odia), Sinhala, Tamil, and Telugu (Johny et al., 2021). In this work, we extend this set by an additional ten Brahmic scripts of South and Maritime South-East Asia: Baybayin (Tagalog), Lepcha, Limbu, Lontara (Bugis), Maithili (Tirhuta), Meetei Mayek, Prachalit (Newa), Sylheti Nagri, Takri and Thaana. These scripts were selected because they are either actively used to record the official regional languages (Vikash and Shubha, 2018; Mohamed, 2019) or are undergoing an active process of revival (Chettri, 2013; Gundayao and Taripe, 2019), often because of their precarious status (Hall et al., 2014).

Extending script coverage to the above set required multiple modifications to akṣara modeling in Nisaba, due to structural features, such as subjoining (dependent) consonants or absence of the vowel suppression sign (*virama*), which were not found in the original ten scripts. Further, Thaana possesses features of both an alphabet and a Brahmic alphasyllabary. The scripts and their respective models are described in Section 3.

Additional mechanisms were also required in Nisaba to support language-specific functions. In Section 4, we will focus on four languages that use the Bengali script: Assamese, Bangla, Kok Borok and Santali.

The original script normalization operations previously provided by Nisaba include both standard Unicode normalization (NFC) and our own visual normalization (Johny et al., 2021). Both of these transformations result in strings that are visually identical to the original. However, in practice one encounters multiple cases when akṣara-level normalization that modifies the visual form is required. Such scenarios include normalization of archaic letters and resolving the ambiguities due to ambiguous or conflicting orthographies. We introduce this type of transformation (denoted *reading* normalization) and discuss it in detail in Section 5.

A final key extension is *fixed-input* transliteration, described in Section 6. The reversible transliteration included in the original library release is based on ISO 15919 standard (ISO, 2001). This scheme is problematic when used for input methods due to difficulties in

[*] On contract from Optimum Solutions, Inc.
[1] https://github.com/google-research/nisaba/

text entry of sequences involving uncommon diacritics such as "ॗ". Fixed-input transliteration fills this gap by providing a mapping from the basic ASCII symbol set to the 8-bit Latin set used by ISO 15919 reversible transliteration, which in turn allows us to unambiguously and economically represent Brahmic scripts in text entry scenarios.

2. Background

Research into natural language processing for South Asian languages has been of increasing interest in recent years, given the evolution of modeling approaches (Jain et al., 2020; Kumar et al., 2020) as well as the improved availability of resources (Roark et al., 2020; Kakwani et al., 2020; Ramesh et al., 2021). Appropriate normalization of these scripts for downstream processing goes beyond what is provided by well-known standards such as NFC. This section offers a brief overview of the core functionality provided by the Nisaba library in relation to Brahmic scripts, described in substantial detail by Johny et al. (2021).

2.1. Structure of Brahmic Scripts

The main unit of linear graphemic representation in Brahmic scripts is known by its traditional Sanskrit-derived name *akṣara*. It is often translated as "syllable" although it does not bear direct correspondence to a syllable of speech, but rather to an orthographic syllable (Bright, 1999). Informally, the structure, or "grammar" of an akṣara is based on the following common principles: an akṣara often consists of a consonant symbol, either bearing, by default, an unmarked *inherent* vowel or, if marked, an attached diacritic (*dependent*) vowel sign; but it may also be an *independent* vowel symbol, or a consonant symbol with its inherent vowel "muted" by a special *virama* diacritic.[2] In any of these preceding scenarios, the base consonant can be replaced by a consonant cluster where all but the last consonant lose their inherent vowel. When the individual component consonants of the cluster combine to form a composite form, precluding the use of an overt *virama* diacritic, this is known as a "consonant conjunct" (Coulmas, 1999; Fedorova, 2013; Share and Daniels, 2016).

2.2. Core Operations

Unicode defines several *normalization forms* which are used for checking whether the two Unicode strings are equivalent to each other (Unicode Consortium, 2021). Nisaba supports Normalization Form C (NFC) which is well suited for comparing visually identical strings (Whistler, 2021). This normalization generally converts strings to the equivalent form that uses composite characters. For example, the legacy sequence *qa* (U+0958) representing Devanagari क़ is canonically equivalent to its NFC form { *ka* (U+0915), *nukta* (U+093C) }.[3]

There are many Unicode character sequences that fall outside the scope of Unicode normalization. Some of these legacy sequences are documented by the Unicode standard or appear in the data mined from the web. To cope with such sequences Nisaba provides *visual normalization* that, in addition to providing the NFC functionality, also supports transforming such legacy sequences. Similar to NFC, this transformation produces visually invariant sequences. Examples of visual normalization include the removal of Devanagari combining vowel sign *e* (U+0947) when it does not affect the compound glyph's visual appearance.[4] Another such example not covered by NFC are the legacy vowel sequences in Gujarati found in the wild, where the letter *au* followed by the vowel sign *ai* is visually equivalent to a single letter *au*.

Nisaba offers *well-formedness validation* that checks whether the given text is readable in a particular script or not. Given the alphasyllabic nature of Brahmic scripts, it would be hard for the non-native reader to visually parse the text if the script rules are not followed. For example, the syllable-final consonants, such as letter *na lonsum* (ꯟ, U+ABDF) in Meetei Mayek, cannot appear word-initially. Well-formedness validation verifies that the text is a sequence of well-formed akṣara using the grammar described above.

The ISO 15919 standard represents a unified 8-bit Latin transliteration scheme for major South Asian Brahmic scripts (ISO, 2001). The Nisaba implementation extended the scheme to accommodate characters not originally covered, e.g., characters added to Unicode after the standard was created, along with adjustments required to provide *reversible transliteration*. Reversibility allows data processing pipelines to use romanized text as an internal representation, then convert it back to the original native script for rendering.

2.3. Finite-state Representation

The Brahmic script structure and manipulation operations described above have a natural interpretation grounded in formal language theory (Sproat, 2000; Sproat, 2003). The akṣara grammar and the corresponding well-formedness validator in Nisaba are represented as finite state automata (FSAs) that correspond to regular languages over Brahmic script character sequences (Yu, 1997). Both the normalization and transliteration operations are represented as weighted finite state transducers (WFSTs) that correspond to regular string-to-string relations (Mohri,

[2] This crude characterization can be complicated by phonological processes, such as schwa deletion (Choudhury et al., 2004).

[3] For ease of reading, when referring to specific symbols, such as DEVANAGARI LETTER KA (क), we will omit the Unicode group when it is clear from context, and present both the letter name in lowercase (here *ka*) and its code (U+0915).

[4] One such example is { ऐ (U+0910), ◌ै (U+0947) } → ऐ (U+0910).

2009). The automata and transducers are compiled offline using Pynini, a Python library for constructing finite-state grammars and for performing operations on WFSTs (Gorman, 2016; Gorman and Sproat, 2021).

3. Script Extensions

The core set of Brahmic scripts supported in the initial release of Nisaba includes ten scripts used for writing the most widely spoken South Asian languages of India, Bangladesh, Nepal and Sri Lanka (Johny et al., 2021). In this work we extend this set by a further ten scripts of South and Maritime South-East Asia. Table 1 shows the entire list of currently supported scripts (additions highlighted in gray) sorted by their ISO 15924 script code (ISO, 2004) along with corresponding script samples. Some of the new additions, such as Meetei Mayek, are in active use, while others, such as Takri, are actively being revived by governments in the region.

We require the minimal set of finite-state operations implemented for a new script to consist of reversible transliteration and well-formedness validation grammars introduced in Section 2. In addition, this set may or may not include further operations such as NFC, visual and reading normalization (introduced in 5 below). The criteria for inclusion of additional operations depends on the script. For example, the reasonably uncomplicated scripts, such as Baybayin and Lontara, may not require any normalization operations at all, while the scripts which are more complex or visually more confusable with the others, such as Tirhuta, require specialized normalization.

3.1. South Asian Brahmic Scripts

Prachalit The Prachalit, or Newa, script is primarily used to write Newar (also called Nepal Bhasa), a Sino-Tibetan language from the Himalayish group spoken in Nepal and the Indian state of Sikkim (Kansakar, 1981; Genetti, 2009). Structurally and visually this script is most similar to Devanagari and also has similarities to the Bengali script, but additionally incorporates necessary Newari language-specific features such as letters for representing resonant breathy consonants (Pandey, 2012).

Sylheti Nagri The Sylheti Nagri script is used to write Sylheti, an Eastern Indo-Aryan language closely related to Bangla. It is spoken in parts of Bangladesh as well as the Assam and Tripura regions of India (Baker et al., 2000; Das, 2017; Simard et al., 2020). Lloyd-Williams et al. (2002) note that while this script has clear Brahmic origins, some features of the script suggest the influence of Perso-Arabic, such as initial lack of a *virama* (*hasanta*) sign, which was only introduced relatively recently. In the present day, its use is not considered obligatory and pronunciations of ambiguous spellings are determined from context.

Takri The Takri script is primarily used to write Dogri, a Northern Indo-Aryan language of Jammu and Kashmir, and neighboring regions in India (Brightbill and Turner, 2007; Kaur and Dwivedi, 2018). The script is considered to be endangered due to the prevalence of Devanagari and the regional governments invest in improving literacy in Takri, for Dogri as well as numerous other threatened languages of the region, such as Jaunsari, that historically it used to record (Pandey, 2009).

Lepcha The Lepcha script is used to write Lepcha, a Sino-Tibetan language from the Himalayish group spoken in parts of Sikkim region of India, Bhutan and Nepal (Plaisier, 2005; Plaisier, 2006), which, according to Campbell and Belew (2018), is severely endangered. Lepcha derives from the Tibetan script and has several interesting features. The script lacks *virama*, which is compensated for by explicit akṣara final consonants that bear no inherent vowel and use of the Tibetan subjoined consonant model. In addition, there is a special sign *ran* (U+1C36) that marks vowel length and accent and can only combine with inherent or dependent vowels and final consonants (Everson, 2005).

Limbu The Limbu script is used to write Limbu, a Himalayish Sino-Tibetan language spoken in eastern Nepal, and the Sikkim and Darjeeling regions of India (van Driem, 1987; Gaenszle, 2021). This language is mentioned as threatened by Campbell and Belew (2018). Similar to Lepcha, this script provides subjoined consonants. In Limbu these are used to indicate "medials" in consonant cluster onsets. A special class of small consonant signs marks syllable-final consonantal positions (bearing no inherent vowel) of native Limbu words, where the other Brahmic scripts would use full consonant letters or *virama*. The additional special feature of Limbu is the dual function of the *sa-i* sign (U+193B) that may indicate vowel lengthening in addition to acting as *virama* (Michailovsky and Everson, 2002).

Meetei Mayek The Meetei Mayek script is used to write Meitei, one of the scheduled languages of India. It is a tonal Sino-Tibetan language from the Kuki-Chin-Naga family and a lingua franca of the Manipur state (Chelliah, 1997; Singh, 2011). Somewhat similar to Limbu, this script uses a special class of explicit silent final consonants in closed syllable codas, but these consonants are represented as full letters rather than combining signs. In modern orthography, the falling tone is sometimes marked with full stop punctuation, whereas in the traditional literature a special *lum iyek* (U+ABEC) sign was used (Everson, 2007).

Maithili The Maithili, or Tirhuta, script is used to write Maithili, an Indo-Aryan language from the Bihari group spoken in the Bihar state of eastern India, where it is one of the scheduled languages, as well as in Nepal (Yadav, 2011; Choudhary, 2013). Visually, Tirhuta bears some superficial similarities with the Bengali script, with seven letters being visually confusable between the two. The major differences concern the formation of the consonant conjuncts and the presence

Name	Id	Sample	Name	Id	Sample
Bengali	`Beng`	বাংলা	Prachalit	`Newa`	𑐥𑑂𑐬𑐔𑐮𑐶𑐟
Lontara	`Bugi`	ᨒᨚᨈᨑ	Oriya	`Orya`	ଓଡ଼ିଆ
Devanagari	`Deva`	देवनागरी	Sinhala	`Sinh`	සිංහල
Gujarati	`Gujr`	ગુજરાતી	Sylheti Nagri	`Sylo`	ꠍꠤꠟꠐꠤ
Gurmukhi	`Guru`	ਗੁਰਮੁਖੀ	Takri	`Takr`	𑚔𑚭𑚊𑚤𑚮
Kannada	`Knda`	ಕನ್ನಡ	Tamil	`Taml`	தமிழ்
Lepcha	`Lepc`	ᰛᰩᰵᰛᰧᰶ	Telugu	`Telu`	తెలుగు
Limbu	`Limb`	ᤕᤠᤰᤌᤢᤱ	Baybayin	`Tglg`	ᜊᜌ᜔ᜊᜌᜒᜈ᜔
Malayalam	`Mlym`	മലയാളം	Thaana	`Thaa`	ތާނަ
Meetei Mayek	`Mtei`	ꯃꯤꯇꯩ ꯃꯌꯦꯛ	Tirhuta	`Tirh`	𑒞𑒱𑒩𑒯𑒳𑒞𑒰

Table 1: Supported scripts sorted by the ISO 15924 script code. The additions are highlighted in gray.

in Tirhuta of vowel signs for short vowels that have no full form equivalents since they cannot occur word-initially (Pandey, 2011).

3.2. Maritime South-East Asian Brahmic Scripts

Baybayin The Baybayin script is used to write Tagalog, a language from Malayo-Polynesian family, which is the official language of the Philippines (Schachter and Reid, 2009; Miller, 2014). Compared to other Brahmic scripts, Baybayin is relatively uncomplicated. No conjunct consonants are formed in this script. The relatively distinct feature of Baybayin is the availability of two vowel muting symbols, *virama* and *pamudpod* (Everson, 1998).

Lontara The Lontara script is used for writing Buginese, Makassarese and Mandar, which are Malayo-Polynesian languages spoken in South Sulawesi province of Indonesia (Grimes and Grimes, 1987; Macknight, 2014). Similar to Baybayin script, Lontara does not record final consonants and does not form consonant conjuncts. In addition, it lacks the traditional *virama*, which complicates the transcription of non-Buginese words that end with a consonant. Various ad hoc modifier marks have been proposed to remedy this (Everson, 2003).

3.3. The Special Case of Thaana

The Thaana script is used in the Maldives to write Dhivehi, an Indo-Aryan language, closely related to Sinhala (Gnanadesikan, 2016). Of all the scripts included so far in the Nisaba library, Thaana is the most distinctive. On the one hand it borrows heavily from the Perso-Arabic abjad. The writing direction is right-to-left. The glyph shapes are similar and several letters such as *sukun* (`U+07B0`), the zero-vowel diacritic, are borrowed directly from Perso-Arabic. However, while the vowels are represented as combining characters, similar to Perso-Arabic, they are always recorded, which makes this system alphabetic. At the same time, since the vowels are diacritics they are clearly subordinate to the consonants in terms of their encoding, which makes this script similar to other alphasyllabaries in the Brahmic family despite the absence of the inherent vowel (De Voogt, 2009). Mohamed (2008) mentions Dhives Akuru, an earlier Brahmic writing system used to record Dhivehi, as a possible influence.

In order to represent Thaana in the Brahmic framework of Nisaba we made a single modification to the finite-state logic which consists of removing all the default transitions dealing with the inherent vowel during construction of the respective FSAs and FSTs. With this modification in place Thaana can be viewed as any other Brahmic system. For example, the *sukun* sign acts as traditional *virama*, while also performing Thaana-specific functions such as marking gemination when preceded by the *noonu* (`U+0782`) letter.

4. Language Extensions

Johny et al. (2021) briefly touched on the language-specific layer of Nisaba, which provides language-specific finite-state operations in addition to script-specific ones. In this section, we detail its use to cover multiple languages using the Bengali script.

According to ScriptSource (Raymond, 2012), the Bengali script is used to write 41 living languages.[5] The initial use of the language layer was to support the two most widely spoken languages that use this script: Assamese and Bangla. Assamese has two extra consonant letters not found in Bangla, the Assamese *ra* (`U+09F0`) and *wa* (`U+09F1`). While the Assamese *wa* is unique, the *ra* letter should not be output by the normalizer together with its Bengali *ra* (`U+09B0`) counterpart. To deal with this we employ N language l specific transducers $\mathcal{T}_i^l$ representing a sequence of context-dependent rewrite rules (Kaplan and Kay, 1994; Mohri and Sproat, 1996) transforming Bengali *ra* into Assamese *ra* (and vice versa) in various contexts. Given a Bengali script-specific normalization transducer $\mathcal{B}$, a language-specific transducer (where l corresponds to either Assamese or Bangla) is obtained from it by a sequence of FST composition operations: $\mathcal{B} \circ \mathcal{T}_1^l \circ \mathcal{T}_2^l \circ \ldots \circ \mathcal{T}_N^l$.

In this work we further extend the set of languages using the Bengali script by two languages. The Bengali script (along with Ol Chiki) is used to write Santali,

[5] `https://www.scriptsource.org`

an Austroasiatic language from the Mundaic group. It is one of the official regional languages of India spoken in several states such as Assam, West Bengal and Tripura (Ghosh, 2008; Choksi, 2018). Most of the Santali adjustments concern the romanization of Santali written in Bengali script, as these should ideally match the corresponding romanization standards of Ol Chiki, which is the official script for Santali. The Bengali script is also used to write Kok Borok, a threatened (Moseley, 2010) Sino-Tibetan language from the Bodo-Garo group spoken in the Indian Tripura state and Bangladesh (Subbarao et al., 2010; Dattamajumdar, 2019). Kok Borok has two additional diphthongs (*vowel letter aw* and *vowel letter ua*) represented as pairs of code-points rather than atomic letters in Unicode (Unicode Consortium, 2021).

For both of these languages, unlike Assamese and Bangla, since letter rewrites are not required, we employ a simpler model, whereby the relevant akṣara component transducers for the Bengali script are expanded (using the FST union operation) with language-specific paths and compiled into a single language-specific transducer. We also follow the same model for providing Hindi-specific operations for the Devanagari script.[6]

5. Reading Normalization

As mentioned in Section 2, the core script normalization utilities in Nisaba are provided in a single finite-state framework consisting of canonical Unicode normalization (NFC) and visual normalization. Visual normalization handles sequences requiring normalization to canonical forms which, although well documented, fall outside the scope of standard Unicode normalization (Johny et al., 2021). The adjective "visual" alludes to the important property of these transformations that in the majority of cases, similar to NFC normalization (Whistler, 2021), they result in canonical forms that render visually identically to the source forms.

As it turns out, visual normalization is not enough to cover all possible script normalization use cases and this is especially true for the Brahmic scripts, for which multiple and/or conflicting orthographic conventions sometimes exist (Iyengar, 2018; Joshi and McBride, 2019). This orthographic variation can result in letter sequences that, although visually distinct, should nevertheless be normalized in some scenarios to a common form. To deal with such cases, in this work we introduce *reading* normalization.

5.1. Implementation

In Nisaba, reading normalization, denoted $\mathcal{R}$, is implemented similarly to visual normalization, denoted $\mathcal{V}$, in terms of FSTs compiled from context-dependent rewrite rules (Kaplan and Kay, 1994; Mohri and Sproat, 1996). A simplified schematic representation of such a transducer $\mathcal{R}$ that handles just one transformation is shown in Figure 1. It implements a simple three-letter context-less rewrite rule for Lepcha retroflex consonants (discussed in detail in Section 5.2 below) using four types of special transitions. Transition labels are in the form $(x : y)$, where x denotes the symbol on the input tape and y on the output tape, respectively. The label x is a shorthand for $(x : x)$.

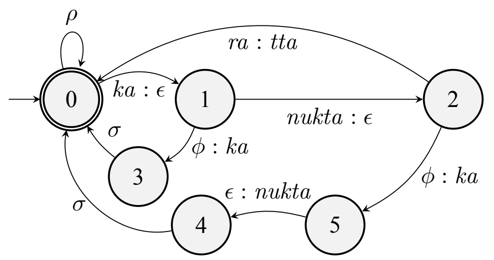

Figure 1: Simplified depiction of an FST corresponding to context-less rewrite of Lepcha retroflex consonant cluster "ᰀᰥ᰷" → "ᱍ" using failure (ϕ, ρ) and exact match (ϵ, σ) transitions. The symbol set Σ in this example consists of Unicode code-points rather than UTF-8 bytes. For $x \in \Sigma^*$, where $x \neq$ ᰀᰥ᰷, this transducer provides an identity mapping. See Section 5.2 on Lepcha, where the need for this transformation is explained.

The ϵ-transitions match anything (including an empty symbol) without consuming the symbol being matched if used on the input tape and produce no output if used on the output tape (Karttunen, 1994). The ϕ-transitions, also known as failure transitions, that leave states 1 and 2 in Figure 1 are traversed only when trying to match with a symbol that does not label any arc leaving that state, an "otherwise" arc (Allauzen et al., 2003). Similar to ϵ-transitions they do not consume any symbols. The σ-transitions that leave states 3 and 4 are similar to ϵ-transitions but consume the input symbol; similarly, the ρ-transitions (a loop-back at state 0) is a symbol-consuming variant of ϕ-transitions (Hall et al., 2015).

The FST architecture allows us to construct low-level script- and language-specific normalization cascades as pictured in Figure 2, which shows a normalization transducer $\mathcal{N} \circ \mathcal{V} \circ \mathcal{R}$ constructed by FST composition from the Unicode NFC, visual and reading normalization FSTs. The resulting three-element pipeline thus constructed is a possible configuration for script normalization that would likely benefit many text processing/modeling scenarios making use of raw native Brahmic script text. There are however certain scenarios where the application of the reading normalization FST $\mathcal{R}$ may not be desirable, such as optical character recognition of historical documents, where visual

[6]Similar to the situation with Bengali, many languages are recorded in Devanagari script, each presenting its orthographic challenges, e.g., Sindhi and Kashmiri (Bhatt, 2015; Iyengar, 2018).

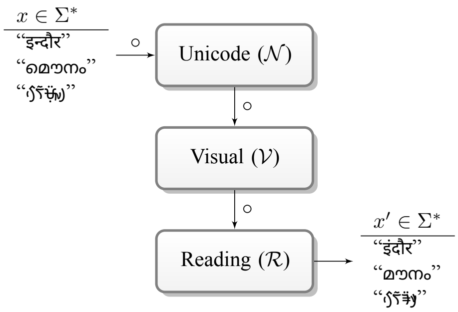

Figure 2: Normalization flow transforming strings x into x' over a particular script Σ using FST composition ($\circ$). Examples for Devanagari, Malayalam, and Lepcha are shown. Components are the Unicode NFC, visual and reading normalization FSTs.

fidelity to the source document would ideally be preserved (Neudecker et al., 2019; Narang et al., 2020; Martínek et al., 2020).

5.2. Examples

Reading normalization examples are plentiful in many languages and scripts in Nisaba, and here we highlight several.

Malayalam The *virama* is known in the Malayalam script as the *candrakkala* sign (U+0D4D), and has a dual function. Similar to other Brahmic scripts, it suppresses the inherent vowel on the preceding consonant, but it is also used to replace the inherent vowel with a neutral vowel sound called *samvruthokaram* (Asher and Kumari, 2012; Johny et al., 2015). In traditional orthography, this is displayed with a *vowel sign u* (U+0D41) followed by *candrakkala*, and in modern orthography (since the 1970s) it is displayed with a *candrakkala* alone, or pseudo-*samvruthokaram* (Chitrajakumar et al., 2005). We bring this orthographic variation into a common form by rewriting { U+0D41, U+0D4D } (◌ു്) sequence as U+0D4D (◌്).

Lepcha There are three retroflex consonants in Lepcha occurring syllable-initially: the voiceless retroflex stop /ʈ/, its apirated version /ʈʰ/ and an unaspirated voiced version /ɖ/. In the traditional script these phonemes do not have corresponding single letters but are recorded using the consonants clusters *kr* (ᰀᰥ᰷), *hr* (ᰝᰥ᰷) and *gr* (ᰃᰥ᰷) that include the Lepcha *nukta* sign (U+1C37), represented as a small dot below the cluster, to distinguish them from the genuine non-retroflex clusters (Plaisier, 2006). In the modern orthography three new individual letters *tta* (U+1C4D), *ttha* (U+1C4E), and *dda* (U+1C4F), were introduced to replace these retroflex clusters (Everson, 2005). In Nisaba, the resulting ambiguity is resolved by rewriting the traditional retroflex encoding to its corresponding single-letter modern form, e.g., mapping { *ga*, *nukta*, *subjoined ra* } (ᰃᰥ᰷) to *dda* (ᱏ), or, as the example in Figure 1 shows, transform { *ka*, *nukta*, *subjoined ra* } (ᰀᰥ᰷) into *tta* (ᱍ) (see Figure 1 for a simplified example of FST representation for this operation).

Hindi Two Devanagari diacritics mark nasalisation processes in Standard Hindi. The Devanagari *anusvāra* (U+0902) is traditionally defined as representing a nasal consonant homorganic (i.e., articulated in the same place of articulation) to a following plosive (Ohala and Ohala, 1991), in contrast to the Devanagari *candrabindu* (or *anunāsika*, U+0901), which marks vowel nasalization and follows a vowel sign, especially at the end of the word (e.g., "माँ"). In practice, however, the two are often used interchangeably in modern writing leading to pronunciation ambiguities (Pandey, 2007). While anusvāra/anunāsika ambiguity cannot be fully resolved purely at the script level without access to the lexical knowledge, we provide a satisfactory normalization of the consonant clusters where the nasal consonants homorganic to the following plosives are recorded explicitly without using *anusvāra*. This is defined for letters $n \in N$ and $s \in S$ as a set of rewrites { n, *virama*, s} $\rightarrow$ { *anusvara*, s}, where N records Hindi nasal consonants and S represents Hindi plosive series. For example, one such rewrite for a bilabial plosive is { *ma* (U+092E), *virama*, *bha* (U+092D) } (म्फ) $\rightarrow$ { *anusvara*, *bha* (U+092D) } (ंफ).

Sylheti The Syloti Nagri script used to record Sylheti has, in addition to the usual dependent vowel diacritics, a special dependent vowel symbol called the *dvisvara* sign (U+A802). This symbol is special because it is used to represent diphthongs with /i/ as a second element when either combining with consonants (forming the diphthong /oi/ with an inherent vowel) or other diphthongs with vowels, both dependent or independent (Das, 2017). Lloyd-Williams et al. (2002) mention an alternative common way to represent these diphthongs in Sylheti by using an independent *letter i* (U+A801) resulting in a spelling ambiguity requiring resolution between *dvisvara* and independent *letter i* when following a consonant.

6. Fixed-input Transliteration

Various input method editors provide transliteration keyboards to help users input language scripts other than Latin. This is useful for regions of the world where there is no prevalent keyboard entry standard for their native language script. Such keyboards often employ statistical models to provide multiple best-fitting candidates for the input the user has entered (Hellsten et al., 2017). This can be excellent for providing accurate native text even when the input is just an approximate phonetic representation of that text. For example, typing "India" would provide the output of the correct Hindi word "इंडिया" in the Devanagari script as the top candidate. However, the word user intended is not always clear because one sequence of Latin text could correspond to multiple intended words in the target language script. For example, the input text "padam" could

Devanagari	Reversible	ITrans
उ	u	u
ऊ	ū	U
अं	aṁ	aM
द	da	da
ड	ḍa	Da
ढ	ḍʰa	Dha
श	śa	sha
ष	ṣa	Sha
स	sa	sa

Table 2: Examples comparing ISO 15919 and fixed-input (ITrans) romanization schemes for the Devanagari script.

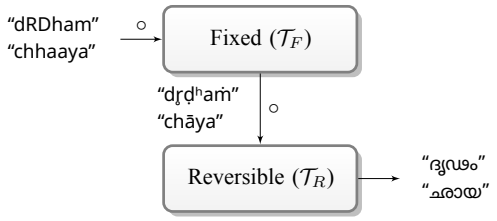

Figure 3: Two-stage transliteration FST pipeline consisting of fixed-input transliteration ($\mathcal{T}_F$) composed with reversible transliteration ($\mathcal{T}_R$). The example strings are in Malayalam.

correspond to many words in the Malayalam script, e.g., "പടം", "പാടം'", "പാടാം", "പാഠം", "പദം", "പാദം", "പതം" or "പഥം". Users may need to select a candidate different from the default.

In order to avoid delays due to candidate selection, some input method editors employ unique phonetic key combinations for all characters in the native script by utilizing case distinctions in the Latin script. For example, to unambiguously represent "इंडिया" users would type "iMDiya". Here the uppercase and lowercase letters are understood as different characters: while "D" represents "ड" (retroflex), lowercase "d" represents "द" (dental). Table 2 shows a list of Devanagari examples highlighting the differences between the ISO 15919-based and fixed-input romanization schemes implemented in Nisaba. Since every Devanagari character is unambiguously represented in this fashion, there is no candidate selection involved and users can type fast as long as they know the rules and follow the correct casing. Because of this, there is a decent user base for such keyboards especially among media professionals. Examples for such schemes are ITrans[7] for Devanagari and Mozhi[8] for Malayalam. Since fixed-input methods do not require training data, they can provide initial transliteration-based input access to scripts in low-resource scenarios.

It is important to note that unlike Nisaba ISO 15919-based transliteration, our implementation of fixed-input transliteration is not reversible because its primary goal is to simplify the text-entry process. Hence, in fixed-input scheme many-to-one mappings are possible, where multiple ASCII input sequences may map to the same representation in native script. Similarly, one-to-many mappings are possible as well. For example, the "kha" fixed-input ASCII sequence maps to a single Devanagari letter *kha* ("ख") or a consonant cluster *ka-virama-ha* ("क्ह").

Implementation: Recall from Section 2.2 that the main (reversible) transliteration scheme implemented in Nisaba is an extension of ISO 15919 that in the forward direction provides conversion from 8-bit Latin character set to native script. To implement an alternative fixed-input transliteration scheme, we provide the additional FST that acts as a lightweight adapter between the fixed-input basic ASCII symbol set and the 8-bit Latin character set required by ISO 15919 transliteration. This adapter is lightweight because the number of required rules is significantly smaller than the number of rules required for a full and reversible ASCII-to-native or 8-bit Latin-to-native script transducer.

Therefore our implementation requires two FSTs for conversion from a fixed-input ASCII character set to native script, as shown in Figure 3 that demonstrates our transliteration pipeline $\mathcal{T} = \mathcal{T}_F \circ \mathcal{T}_R$ consisting of the fixed-input transliterator $\mathcal{T}_F$ composed with the ISO 15919-based transliterator $\mathcal{T}_R$. Given an ASCII sequence x, transliteration into the corresponding Brahmic script is obtained by the sequence of FST compositions $(x \circ \mathcal{T}_F) \circ \mathcal{T}_R$.

7. Conclusion and Future Work

In this paper we provided an overview of substantial extensions to the finite-state Brahmic script processing library, both in terms of improved script and language coverage, as well as additional finite-state utilities. The expansion into lesser-used Brahmic scripts poses its technical challenges and we described our approach to these using an akṣara-centered finite-state design. We presented a new type of *reading* script normalization, which, in addition to the visual fidelity-preserving normalizations, provides a script-level utility for reducing orthographic ambiguities. Finally, we introduced *fixed-input* transliteration that simplifies text entry in Brahmic scripts when used in conjunction with reversible transliteration.

It is worth noting that while the new set of scripts and languages covered in this paper required some modifications to our finite-state framework, such as relaxing the inherent vowel requirement to accommodate Thaana or introducing the support for subjoined consonants for scripts from Tibetan family, these modifications were relatively minor and the current frame-

[7] https://www.aczoom.com/itrans/

[8] https://sites.google.com/site/cibu/mozhi/mozhi2

work is flexible enough to accommodate for other Brahmic scripts of similar structural complexity, such as Kaithi. There are, however, a significant number of Brāhmī-derived scripts that we leave for future work, such as Burmese and Thai, which are structurally different enough from the ones we dealt with so far in Nisaba to require more substantial design changes of our algorithms.

At the time of writing, the amount of Unicode-encoded text available on the web in many of the scripts for low-resource languages we described (such as Limbu and Takri) is rather limited. This is mostly due to their recent introduction into the Unicode standard compared to the major Brahmic scripts, but also due to other confounding factors such as script illiteracy or lack of standard orthographies. Furthermore, some data available in these scripts was generated via simple automatic rule-based transliteration from a dominant script (such as Devanagari), a process that often disregards target script peculiarities. We expect to continue improving library support for such scripts and related low-resource and endangered languages in Nisaba as online data availability grows through community- and regional government-based data digitization and script revival efforts.

8. Acknowledgments

The authors would like to thank Anna Katanova for her help with this project and the anonymous reviewers for useful feedback on the earlier draft of this paper.

9. Bibliographical References

Allauzen, C., Mohri, M., and Roark, B. (2003). Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 40–47, Sapporo, Japan, July. Association for Computational Linguistics.

Asher, R. E. and Kumari, T. C. (2012). *Malayalam*. Descriptive Grammars. Routledge, London & New York.

Baker, J., Lie, M., McEnery, A., and Sebba, M. (2000). Building a corpus of spoken Sylheti. *Literary and Linguistic Computing*, 15(4):421–432.

Bhatt, R. M. (2015). Script choice, language loss and the politics of anamnesis: Kashmiri in diaspora. In Christopher Stroud et al., editors, *Language, Literacy and Diversity*, Routledge Critical Studies in Multilingualism, pages 130–147. Routledge.

Bright, W. (1999). A matter of typology: Alphasyllabaries and abugidas. *Written Language & Literacy*, 2(1):45–55.

Brightbill, J. D. and Turner, S. D. (2007). A sociolinguistic survey of the Dogri language, Jammu and Kashmir. SIL Electronic Survey Report 2007-017, Journal of Language Survey Reports, SIL International, July.

Campbell, L. and Belew, A. (2018). *Cataloguing the World's Endangered Languages*. Routledge, London & New York.

Chelliah, S. L. (1997). *A Grammar of Meithei*, volume 17 of *Mouton Grammar Library [MGL]*. Mouton de Gruyter, Berlin, Germany.

Chettri, M. (2013). *Ethnic politics in the Nepali public sphere: Three cases from the Eastern Himalaya*. Ph.D. thesis, School of Oriental and African Studies (SOAS), University of London. Department of South Asia, Faculty of Languages and Cultures.

Chitrajakumar, R., Gangadharan, N., and Vedi, R. A. (2005). Problems of Malayalam encoding in the Indic context. Technical Report L2/05-308, Unicode Consortium.

Choksi, N. (2018). Script as constellation among Munda speakers: the case of Santali. *South Asian History and Culture*, 9(1):92–115.

Choudhary, P. (2013). Causes and effects of super-stratum language influence, with reference to Maithili. *Journal of Indo-European studies*, 41(3/4):378–391.

Choudhury, M., Basu, A., and Sarkar, S. (2004). A diachronic approach for schwa deletion in Indo Aryan languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 20–26, Barcelona, Spain, July. Association for Computational Linguistics.

Coulmas, F. (1999). *The Blackwell Encyclopedia of Writing Systems*. John Wiley & Sons, Oxford.

Das, A. R. (2017). *A Comparative Study of Bangla and Sylheti Grammar*. Ph.D. thesis, Universita degli Studi di Napoli "Federico II", Naples, Italy, October. Dipartimento di Studi Umanistici.

Datta, A. K. (1984). A generalized formal approach for description and analysis of major Indian scripts. *IETE Journal of Research*, 30(6):155–161.

Dattamajumdar, S. (2019). A brief history of linguistic science with special reference to the Bodo, Garo and Kokborok languages of North-East India. *Indian Journal of History of Science*, 54:69–89.

De Voogt, A. (2009). Languages and scripts in the Maldive Islands: Coding and encoding. In Alex De Voogt et al., editors, *The Idea of Writing: Play and Complexity*, volume 1 of *The Idea of Writing*, pages 197–205. Brill.

Everson, M. (1998). Revised proposal for encoding the Philippine scripts in the UCS. ISO/IEC JTC1/SC2/WG2 N1933, Unicode Consortium, November.

Everson, M. (2003). Revised final proposal for encoding the Lontara (Buginese) script in the UCS. ISO/IEC JTC1/SC2/WG2 N2633R, Unicode Consortium, October.

Everson, M. (2005). Proposal for encoding the Lepcha script in the BMP of the UCS. ISO/IEC

JTC1/SC2/WG2 N2947R, Unicode Consortium, July.

Everson, M. (2007). Proposal for encoding the Meitei Mayek script in the BMP of the UCS. ISO/IEC JTC1/SC2/WG2 N3206R2, Unicode Consortium, August.

Fedorova, L. L. (2013). The development of graphic representation in abugida writing: The akshara's grammar. *Lingua Posnaniensis*, 55(2):49–66.

Gaenszle, M. (2021). The Limbu script and the production of religious books in Nepal. *Philological Encounters*, 6(1-2):43–69.

Genetti, C. (2009). *A grammar of Dolakha Newar*, volume 40 of *Mouton Grammar Library [MGL]*. Mouton de Gruyter, Berlin, Germany.

Ghosh, A. (2008). Santali. In Gregory D. S. Anderson, editor, *The Munda Languages*, volume 3 of *Routledge Language Family Series*, pages 11–98. Routledge, London & New York.

Gnanadesikan, A. E. H. (2016). *Dhivehi: The Language of the Maldives*, volume 3 of *Mouton-CASL Grammar Series*. Mouton de Gruyter, Berlin, Germany.

Gorman, K. and Sproat, R. (2021). *Finite-State Text Processing*, volume 14 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Gorman, K. (2016). Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80, Berlin, Germany, August. Association for Computational Linguistics.

Grimes, C. E. and Grimes, B. D. (1987). *Languages of South Sulawesi*, volume 38 of *Materials in Languages of Indonesia*. Dept. of Linguistics, Research School of Pacific Studies, The Australian National University.

Gundayao, B. C. and Taripe, R. B. (2019). Baybayin and the proposal for a national writing system: Knowledge and attitude among university students in Quezon City, Philippines. In Michael Kho Lim et al., editors, *BILANGAN 2: Selected Papers from the 2019 International Conference on Cultural Statistics and Creative Economy*, pages 62–72. National Commission for Culture and the Arts, Manila, Philippines.

Hall, P., Bal, B. K., Dhakhwa, S., and Regmi, B. N. (2014). Issues in encoding the writing of Nepal's languages. In *15th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 52–67, Kathmandu, Nepal, April. Springer.

Hall, K., Cho, E., Allauzen, C., Beaufays, F., Coccaro, N., Nakajima, K., Riley, M., Roark, B., Rybach, D., and Zhang, L. (2015). Composition-based on-the-fly rescoring for salient n-gram biasing. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1418–1422, Dresden, Germany, September. International Speech Communication Association.

Hellsten, L., Roark, B., Goyal, P., Allauzen, C., Beaufays, F., Ouyang, T., Riley, M., and Rybach, D. (2017). Transliterated mobile keyboard input via weighted finite-state transducers. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, pages 10–19, Umeå, Sweden, September. Association for Computational Linguistics.

ISO. (2001). ISO 15919: Transliteration of Devanagari and related Indic scripts into Latin characters. `https://www.iso.org/standard/28333.html`. International Organization for Standardization, Geneva, Switzerland.

ISO. (2004). ISO 15924: Codes for the representation of names of scripts. `https://www.iso.org/obp/ui/#iso:std:iso:15924:ed-1:v1:en`. International Organization for Standardization, Geneva, Switzerland.

Iyengar, A. (2018). Variation in Perso-Arabic and Devanāgarī Sindhī orthographies: An overview. *Written Language & Literacy*, 21(2):169–197.

Jain, K., Deshpande, A., Shridhar, K., Laumann, F., and Dash, A. (2020). Indic-Transformers: An analysis of transformer language models for Indian languages. *arXiv preprint arXiv:2011.02323*.

Johny, C., Alex, S., and S., S. V. (2015). Proposal to encode Malayalam Sign Circular Virama. ISO/IEC JTC1/SC2/WG2 L2/14-014R, Unicode Consortium, January.

Johny, C., Wolf-Sonkin, L., Gutkin, A., and Roark, B. (2021). Finite-state script normalization and processing utilities: The Nisaba Brahmic library. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 14–23, Online, April. Association for Computational Linguistics.

Joshi, R. M. and McBride, C. (2019). *Handbook of Literacy in Akshara Orthography*, volume 17 of *Literacy Studies*. Springer, Switzerland.

Kakwani, D., Kunchukuttan, A., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M. M., and Kumar, P. (2020). IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online, November. Association for Computational Linguistics.

Kansakar, T. R. (1981). Newari language and linguistics: Conspectus. *Contributions to Nepalese Studies*, 8(2):1–18.

Kaplan, R. M. and Kay, M. (1994). Regular models of

phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Karttunen, L. (1994). Constructing lexical transducers. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 1, pages 406–411, Kyoto, Japan, August.

Kaur, K. and Dwivedi, A. V. (2018). *Dogri and its Dialects: A Comparative Study of Kandi and Pahari Dogri*, volume 115 of *Linguistics Edition*. Lincom Europa Academic Publications, Munich, Germany.

Kumar, S., Kumar, S., Kanojia, D., and Bhattacharyya, P. (2020). "A passage to India": Pre-trained word embeddings for Indian languages. In *Proc. of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 352–357, Marseille, France, May. European Language Resources association.

Lloyd-Williams, J., Lloyd-Williams, S., and Constable, P. (2002). Documentation in support of proposal for encoding Syloti Nagri in the BMP. ISO/IEC JTC1/SC2/WG2 L2/02-388, Unicode Consortium, November.

Macknight, C. (2014). The triumph of Lontara'. In *Proceedings of International Workshop on Endangered Scripts of Island Southeast Asia*, pages 1–22, Research Institute for Languages and Cultures of Asia and Africa, Tokyo University of Foreign Studies, February.

Martínek, J., Lenc, L., and Král, P. (2020). Building an efficient OCR system for historical documents with little training data. *Neural Computing and Applications*, 32(23):17209–17227.

Michailovsky, B. and Everson, M. (2002). Revised proposal to encode the Limbu script in the UCS. ISO/IEC JTC1/SC2/WG2 N2410, Unicode Consortium, February.

Miller, C. (2014). A survey of indigenous scripts of Indonesia and the Philippines. In *Proceedings of International Workshop on Endangered Scripts of Island Southeast Asia*, pages 1–49, Research Institute for Languages and Cultures of Asia and Africa, Tokyo University of Foreign Studies, February.

Mohamed, N. (2008). Scripts of Maldives: Eveyla Akuru, Dhives Akuru and Thaana. In *Essays on Early Maldives*. National Centre of Linguistic and Historical Research (NCLHR), Male', Maldives, 2nd edition.

Mohamed, N. (2019). From a monolingual to a multilingual nation: Analysing the language education policy in the Maldives. In *The Routledge International Handbook of Language Education Policy in Asia*, pages 414–426. Routledge.

Mohri, M. and Sproat, R. (1996). An efficient compiler for weighted rewrite rules. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 231–238, Santa Cruz, California, USA, June. Association for Computational Linguistics.

Mohri, M., Pereira, F., and Riley, M. (2000). The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mohri, M. (2009). Weighted automata algorithms. In Manfred Droste, et al., editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, pages 213–254. Springer.

Moseley, C. (2010). *Atlas of the World's Languages in Danger*. UNESCO Publications Office, Paris, France, 3rd edition.

Narang, S. R., Jindal, M. K., and Kumar, M. (2020). Ancient text recognition: a review. *Artificial Intelligence Review*, 53(8):5517–5558.

Neudecker, C., Baierer, K., Federbusch, M., Boenig, M., Würzner, K.-M., Hartmann, V., and Herrmann, E. (2019). OCR-D: An end-to-end open source OCR framework for historical printed documents. In *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, pages 53–58, Brussels, Belgium, May.

Ohala, M. and Ohala, J. J. (1991). Nasal epenthesis in Hindi. *Phonetica*, 48(2-4):207–220.

Pandey, P. (2007). Phonology–orthography interface in Devanāgarī for Hindi. *Written Language & Literacy*, 10(2):139–156.

Pandey, A. (2009). Proposal to encode the Takri script in ISO/IEC 10646. ISO/IEC JTC1/SC2/WG2 N3758, Unicode Consortium, December.

Pandey, A. (2011). Proposal to encode the Tirhuta script in ISO/IEC 10646. ISO/IEC JTC1/SC2/WG2 N4035, Unicode Consortium, May.

Pandey, A. (2012). Proposal to encode the Newar script in ISO/IEC 10646. ISO/IEC JTC1/SC2/WG2 N4184, Unicode Consortium, February.

Plaisier, H. (2005). A brief introduction to Lepcha orthography and literature. *Bulletin of Tibetology*, 41(1):7–24.

Plaisier, H. (2006). *A Grammar of Lepcha*. Ph.D. thesis, University of Leiden, the Netherlands, February.

Ramesh, G., Doddapaneni, S., Bheemaraj, A., Jobanputra, M., AK, R., Sharma, A., Sahoo, S., Diddee, H., J, M., Kakwani, D., Kumar, N., Pradeep, A., Deepak, K., Raghavan, V., Kunchukuttan, A., Kumar, P., and Khapra, M. S. (2021). Samanantar: The largest publicly available parallel corpora collection for 11 Indic languages. *arXiv preprint arXiv:2104.05596*.

Raymond, M. (2012). ScriptSource: Making information on the world's scripts and languages accessible. In *Charting Vanishing Voices: A Collaborative Workshop to Map Endangered Oral Cultures*, Cambridge, UK, June. Presentation.

Roark, B., Wolf-Sonkin, L., Kirov, C., Mielke, S. J., Johny, C., Demirsahin, I., and Hall, K. (2020). Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2413–2423.

Schachter, P. and Reid, L. A. (2009). Tagalog. In

Bernard Comrie, editor, *The World's Major Languages*, pages 844–866. Routledge, 2nd edition.

Share, D. L. and Daniels, P. T. (2016). Aksharas, alphasyllabaries, abugidas, alphabets and orthographic depth: Reflections on Rimzhim, Katz and Fowler (2014). *Writing Systems Research*, 8(1):17–31.

Simard, C., Dopierala, S. M., and Thaut, E. M. (2020). Introducing the Sylheti language and its speakers, and the SOAS Sylheti project. *Language Documentation and Description*, 18:1–22.

Singh, H. T. (2011). The evolution and recent development of the Meitei Mayek script. In *North East Indian Linguistics*, volume 3, pages 24–32. Cambridge University Press India, New Delhi, India.

Sproat, R. (2000). *A Computational Theory of Writing Systems*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Sproat, R. (2003). A formal computational analysis of Indic scripts. In *In International Symposium on Indic Scripts: Past and Future*, Tokyo, Japan, December.

Subbarao, K. V., Malhotra, S., and Barua, S. (2010). The structure of the Kokborok language. *Interdiciplinary Journal of Linguistics (Kashmir)*, III:1–43.

Unicode Consortium. (2021). The Unicode Standard. Online: `http://www.unicode.org/versions/Unicode14.0.0`. Version 14.0.0, Mountain View, CA.

van Driem, G. (1987). *A grammar of Limbu*, volume 4 of *Mouton Grammar Library [MGL]*. Mouton de Gruyter, Berlin, Germany.

Vikash, B. and Shubha, H. (2018). Effect of shifting orthographic practices of Manipuri Script on millennials. *Global Media Journal: Indian Edition*, 10(2).

Whistler, K. (2021). Unicode normalization forms. Technical Report TR15-51, Unicode Consortium, August. Version 14.0.0.

Yadav, R. (2011). *A reference grammar of Maithili*, volume 11 of *Trends in Linguistics. Documentation [TiLDOC]*. Mouton de Gruyter, Berlin, Germany.

Yu, S. (1997). Regular languages. In Grzegorz Rozenberg et al., editors, *Handbook of Formal Languages*, volume 1: Word, Language, Grammar, pages 41–110. Springer, Berlin.