Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 6662–6673
Marseille, 20-25 June 2022

Criteria for Useful Automatic Romanization in South Asian Languages

Işın Demirşahin[†], **Cibu Johny**[†], **Alexander Gutkin**[†], **Brian Roark**[‡]
Google Research
[†]United Kingdom [‡]United States
{isin,cibu,agutkin,roark}@google.com

Abstract

This paper presents a number of possible criteria for systems that transliterate South Asian languages from their native scripts into the Latin script, a process known as romanization. These criteria are related to either fidelity to human linguistic behavior (pronunciation transparency, naturalness and conventionality) or processing utility for people (ease of input) as well as under-the-hood in systems (invertibility and stability across languages and scripts). When addressing these differing criteria several linguistic considerations, such as modeling of prominent phonological processes and their relation to orthography, need to be taken into account. We discuss these key linguistic details in the context of Brahmic scripts and languages that use them, such as Hindi and Malayalam. We then present the core features of several romanization algorithms, implemented in a finite state transducer (FST) formalism, that address differing criteria. Implementations of these algorithms have been released as part of the Nisaba finite-state script processing library.

Keywords: transliteration, romanization, South Asian languages, Brahmic scripts

1. Introduction

Transliteration is the conversion of language represented in one script to the same language represented in another script (Wellisch, 1978). For example, the Russian word “гласность” is most often transliterated into the Latin script as “glasnost”. While translation involves a change in language – e.g., “гласность” becomes “transparency” when translated to English – transliteration, in contrast, exactly preserves the linguistic content, i.e., “glasnost” is still a Russian word (just written in a different script). Beyond isolated words or names transliterated in the news, whole sentences can be transliterated, such as “idet sneg” which is Russian for “it is snowing”, which would normally be written “идет снег” in the Cyrillic script.

Romanization is the special case of transliteration where the target script is the Latin script. It is not a special case due to any particular characteristics of the transliteration problem when the Latin alphabet is involved; rather because it is so common. It is particularly common in South Asia, where many factors have caused romanization of these languages to be ubiquitous. Despite this ubiquity, there is generally no standard orthography for South Asian languages in the Latin script, leading to wide variability.

When automatically romanizing South Asian languages, the “right” choice often depends heavily on the use case. In this paper, we present a number of distinct criteria for producing romanizations, and describe several approaches to providing romanizations that satisfy these (sometimes conflicting) criteria across a number of languages and scripts. Briefly, these criteria are: invertibility; pronunciation transparency; ease of input; naturalness; conventionality; and stability across languages and scripts — see Section 3 for a full description of these criteria. No romanization system can fulfill all of these criteria, but a particular use case, such as presenting a phrase to be read aloud, may favor certain criteria over others (e.g., pronunciation transparency), and different romanization systems can be used to support these varied scenarios.

After presenting background on romanization in South Asia and related work in automatic transliteration, we will go into depth on our identified criteria with examples from existing romanization standards and text corpora. We will then present several algorithms implemented in the Nisaba library[1] (Johny et al., 2021), with a particular focus (for ease of exposition) on Hindi and Malayalam.

2. Background

2.1. Romanization in South Asia

The documented history of romanization in South Asia begins with the early work of Jesuit Catholic missionaries in late 16th century in South India (Veliath, 2011; Amaladass, 2017; Mahboob and Rahman, 2017; Flüchter and Nardini, 2020) and Bengal (Chakrabortty and Chakrabortty, 1976; Mahboob and Rahman, 2017). Dealing with local languages, such as Konkani, Marathi, Tamil and Bengali, the missionaries often resorted to transcribing these languages in the Latin script due to the lack of types for native scripts, such as Devanagari (Veliath, 2011). Following the establishment of British colonial rule in the late 18th century, many romanization strategies for Brahmic and Perso-Arabic scripts were proposed, but none of them, including the Hunterian system endorsed by the governments in India, Bangladesh and Pakistan, gained wide acceptance (Iyengar, 2015).

[1]`https://github.com/google-research/nisaba/`

In the early years of machine readable text, poor encoding and font support led to the widespread use of the Latin script for text input in languages natively using other scripts, even those using alphabets such as Cyrillic (Jones, 1975; Sen and Sur, 1979; Sinha and Srinivasan, 1984; Mukhopadhyay et al., 1985). In the current era of widespread mobile computing, virtual keyboards for alphabets are relatively straightforward, but other writing systems are often input using romanization systems such as Pinyin for Chinese (Li and Li, 2019). In South Asia, the Brahmic scripts are challenging to input directly in mobile text entry (see e.g., Hellsten et al., 2017), yet these languages lack a common standard romanization system; even so a significant portion of on-line text in these languages is written in the Latin script (Pavan et al., 2010; Krishnan et al., 2021).

2.2. Automatic Transliteration

Machine transliteration is a well-studied area. Early work on automatic transliteration between writing systems (mostly East Asian, Perso-Arabic and Latin scripts) was driven by the needs of statistical machine translation or information retrieval systems, and hence was generally focused on proper names and/or loanwords (Arbabi et al., 1994; Knight and Graehl, 1998; Chen et al., 1998; Wan and Verspoor, 1998; Jung et al., 2000; Al-Onaizan and Knight, 2002; Virga and Khudanpur, 2003; Li et al., 2004). Antony and Soman (2011) provide an overview of the early statistical transliteration systems for major Indian languages and offer a crude taxonomy of statistical approaches dividing them into pure grapheme-based (Lee and Choi, 1998), those utilizing phonological knowledge (Knight and Graehl, 1998), and hybrid, or correspondence-based, models (Al-Onaizan and Knight, 2002). Additional more comprehensive surveys on statistical methods for machine transliteration are provided by Karimi et al. (2011), and Prabhakar and Pal (2018). The progress in this area, especially in data-intensive methods reviewed below, has been hindered by a relative scarcity of transliteration corpora for South Asian languages, although this situation has been gradually improving in recent years (Bhat et al., 2014; Khapra et al., 2014; Kunchukuttan et al., 2015; Roark et al., 2020).

With recent advances in neural methods for NLP (Conneau et al., 2020) and increased availability of resources for South Asian languages, both in terms of corpora and pretrained models (Kakwani et al., 2020; Khanuja et al., 2021), there has been a renewed interest in romanization techniques. This is primarily driven by recent successes in multilingual neural language modeling and neural machine translation (NMT), where romanization is generally beneficial as a mechanism for unifying multi-script training and adaptation data (Chakravarthi et al., 2019; Chakravarthi et al., 2020; Datta et al., 2020; Amrhein and Sennrich, 2020; Zhang et al., 2020; Khatri et al., 2021; Appicharla et al., 2021), especially for related languages (Muller et al., 2021). In addition, romanization was shown to benefit diverse downstream multilingual NLP tasks, such as morphological analysis (Hauer et al., 2019; Murikinati et al., 2020), named entity recognition (Huang et al., 2019) and part-of-speech tagging (Cardenas et al., 2019).

In the above-mentioned NLP systems, the romanization component is most often rule-based, implemented using one of the popular transliteration libraries (Hermjakob et al., 2018; Kunchukuttan, 2020; Rajan, 2020). One reason for this is related to the *invertibility* criterion that we mentioned earlier. Invertible romanization systems avoid information loss during romanization, a consideration that has been shown to be beneficial for some multilingual methods (Amrhein and Sennrich, 2020). Such requirements are easier to satisfy with rule-based approaches than with learned systems.

There is also active work on transliteration per se investigating learned sequence-to-sequence modeling approaches (Patel et al., 2020; Kunchukuttan et al., 2021; Ryskina et al., 2021). In the context of back transliteration (or, *deromanization*), recent work includes efficient finite state-based techniques for input methods (Hellsten et al., 2017; Wolf-Sonkin et al., 2019); back transliteration of informal code-mixed text (Riyadh and Kondrak, 2019); and noisy channel methods modeling both phonetic and visual orthographic similarity (Ryskina et al., 2020). The bulk of this work is focused on naturally occurring romanized text, hence simuluated training data for such tasks (one use for automatic romanization) would focus on the naturalness and conventionality criteria rather than invertibility.

In this paper, we focus on algorithms for romanization based on explicit grammars compiled into finite-state transducers, rather than learned models of romanization. As mentioned above, such approaches can provide useful benefits to downstream modeling tasks, e.g., those that perform better with lossless (or less lossy) transliterations. They can also provide a starting point for modeling in the face of data sparsity, which is a consideration for many of the South Asian languages included in the Nisaba library (see next section). Ultimately, hybrid solutions that both encode linguistic knowledge and learn from whatever data is available will likely provide the best results. The algorithms in the current paper provide a basis for exploring such approaches.

2.3. Nisaba Library

The Nisaba library (Johny et al., 2021) is a collection of utilities for performing low-level script processing for South Asian scripts, particularly Brahmic scripts. These operations include visual normalization, whereby legacy Unicode encodings are converted to their visually-indistinguishable canonical forms; validity checks to ensure that strings are legible; and reversible transliteration. The library uses OpenFst (Allauzen et al., 2007) and Pynini (Gorman, 2016) to provide easy-to-interpret script- and language-specific resources that are efficiently compiled into finite-state transducers for processing strings. Recent updates to the library (Gutkin et al., 2022) extend functionality to many more scripts and languages, as well as adding a second form of reversible transliteration that we will discuss in the next section.

3. Criteria for Useful Romanization

In this section, we discuss at greater length the diverse (but not necessarily exhaustive) criteria that we have identified as impacting the utility of romanization in various scenarios: invertibility; pronunciation transparency; ease of input; naturalness; conventionality; and stability across languages and scripts.

Invertibility of romanization means that the result can be transliterated back to the exact original input string. With some minor exceptions, this a characteristic of the ISO 15919 standard (ISO, 2001), which maps between Brahmic scripts and the Latin script in both directions. For example, the Hindi word अस्पताल is romanized in ISO 15919[2] as "aspatāla". This is invertible because the extended Latin script string directly encodes the individual Unicode codepoints from the original Devanagari string[3]: 'a' represents Devanagari Letter A (`U+0905`: अ); 'spa' represents the letter SA (`U+0938`: स), followed by the vowel canceling sign Virama (`U+094d`: ्) and the letter PA (`U+092a`: प); 'tā' represents the letter TA (`U+0924`: त) followed by the vowel sign AA[4] (`U+093e`: ा); and finally 'la' represents the letter LA (`U+0932`: ल). Invertibility is beneficial, for example, in situations where it is preferable to process the Latin script string system-internally for whatever reason, while preserving the ability to output the result in the original script. Examples of such scenarios might include calculation of some kind of baseline edit distance between tokens written in different scripts.

The Nisaba library includes a version of reversible ISO 15919 transliteration, extended to include Unicode characters added after the standard was created. Recent extensions to the library (Gutkin et al., 2022) add a second reversible transliteration for a subset of the languages, which differs from ISO 15919 in using only basic ASCII characters on the Latin script side.

As convenient as these invertible representations are, their **pronunciation transparency** is generally low. In the particular example above, the word final vowel is not pronounced (known as *schwa deletion*), something that is not explicitly marked with virama. A more pronunciation-transparent romanization would be "aspatāl". In general, however, pronunciation transparency and invertibility will conflict, since multiple words written distinctly in the native script will share pronunciations. Alternative pronunciation-transparent romanizations restricted to the basic Latin script without diacritics — such as "aspataal" in the current case — may be preferred due to **ease of input**. This latter criterion of input ease was a key rationale for the second reversible transliteration method mentioned above, that uses only basic ASCII.

Naturalness and conventionality are criteria that may move romanization away from pronunciation transparency. **Naturalness** refers to the way speakers of the language tend to spontaneously romanize words. For example, individuals may or may not choose to represent long vowels by doubling the vowel in the Latin script, which would lead to the above example being romanized as "aspatal" with neither doubling nor diacritics to indicate the long vowel. General tendencies regarding explicit marking of processes like consonant and vowel doubling or aspiration can move away from perfect correspondence to pronunciation, and the resulting variation is often present in human produced romanizations such as those provided in the Dakshina dataset (Roark et al., 2020). **Conventionality** may apply for a subset of words with a standard spelling, e.g., for English loan words. For the current example, despite the absence of initial aspiration (as in some English dialects), individuals may choose to romanize the word as "hospital". Both naturalness and conventionality influence how people spontaneously romanize and must be accounted for by any automatic methods that attempt to be guided by actual human linguistic behavior.

Finally, **stability across languages and scripts** refers to romanizations of the same word from different scripts. In certain processing scenarios, one might prefer that the same word written in different scripts (e.g., proper names, loan words, etc.) would yield the same romanization, so that it can be seen to be the same word. For example, multilingual speech recognition training for languages using multiple scripts can be achieved by romanizing the transcripts from all of the languages, training the models, then transliterating back into the original script after the recognizer transcribes the speech (Datta et al., 2020). Generalization across languages will occur when the same words are represented similarly across the languages. For example, the Bengali word হাসপাতাল yields an ISO 15919 romanization of "hāsapātāla", and would have pronunciation-transparent and/or natural romanizations of "haaspaataal" or "haspatal", none of which exactly match the Hindi version of the loan word. Only the conventional English spelling of "hospital" would yield identity across the languages. Given the prevalence of common loan words through the languages of South Asia, this is a tricky condition yet potentially important for effective multilingual language processing.

[2]As implemented in the Nisaba library (Johny et al., 2021): `https://github.com/google-research/nisaba/`.

[3]There is not always a one-to-one correspondence between symbols, hence some context may be required to disambiguate between alternatives.

[4]The ā symbol can also represent an independent long vowel in other contexts.

4. Multiple Criteria Brahmic Script Romanization

In this section we present an approach to romanization of Brahmic scripts that attends to several of the above criteria – particularly pronunciation transparency, naturalness, conventionality and stability across languages and scripts. Before jumping into the implementation specifics, we cover some key linguistic considerations required to adequately address common phenomena in the scripts. Our approach maps from the ISO 15919 reversible romanization to one of several alternative outputs via a common internal representation that allows for pronunciation-, naturalness- and convention-driven operations.

4.1. Key Linguistic Details

For romanization, we apply a variety of operations to an input ISO 15919 string that are driven either by pronunciation or language- or task-specific transliteration conventions. The most prominent phonological operations that are reflected in romanization are handling of the inherent vowels, place assimilation of nasal markers, and voicing.

Inherent vowels: In Brahmic scripts, consonant symbols bear an inherent vowel (schwa), which can be overridden by a dependent vowel sign attached to the consonant, or deleted by a virama (Bright, 1999). Some scripts do not always explicitly mark the inherent vowel deletion, whereas others might have separate symbols for a subset of vowels without inherent vowels. For example, Hindi, Malayalam, and Telugu handle the consonant cluster in the word "Farsi" in three different ways. The Telugu word "ఫార్సీ" (p^{h}ārsī) uses the explicit deletion marker virama "◌్" to indicate that there is no inherent vowel after the letter "ra" "ర". The Malayalam word "ഫാർസി" (p^{h}ār̲ˀsi) uses the "chillu rr" symbol "ർ" which does not bear an inherent vowel. The Hindi word "फ़ारसी" "fārasī" does not have any explicit markers for schwa deletion for the Devanagari letter "ra" "र". The Hindi case requires the reader to either know the word, or have a heuristic for deciding whether an inherent vowel is pronounced or not (Pandey, 1990). See Section 4.2 for specific details on how we handle this problem.

Nasal assimilation: Nasal markers such as *anusvara* and *candrabindu* are assimilated to the place of articulation of the following consonant in many languages using Brahmic scripts, such as Assamese (Dutta, 2019, p. 187) and Hindi (Pandey, 2007). Assimilation can happen at a wide range of places of articulation such as labial, dental, alveolar, velar and retroflex, depending on the language, and romanized as labial "m" or non-labial "n". For example, in Hindi "चंदा" (caṁdā) is typically romanized as "chanda" while "चंबा" (caṁbā) is usually romanized as "chamba". When there is no following consonant, the nasal marker anusvara can indicate a nasalized vowel and transliterated as "n", for example in the Hindi word "आयातों" (āyātoṁ) "ayaton". Or it can be pronounced and transliterated as a consonant, such as "m" as in Malayalam "അക്കം" (akkaṁ) "akkam", Telugu "అంగం" (aṁgaṁ) "angam" and Kannada "ಕಲಂ" (kalaṁ) "kalam".

Voicing: In languages like Malayalam and Tamil, voiceless stops can be voiced in certain contexts (Asher and Kumari, 2012; Annamalai and Steever, 2019). For example, in Malayalam "കടൽ" (kaṭal) "kadal", the retroflex "ṭ" is voiced in the intervocalic position and it is transliterated as "d". In Tamil "தூங்க" (tūṅka) "thoonga" the "k" after the nasal is voiced and transliterated as "g" when followed by a vowel.

Additional adjustments: There are additional pronunciation-driven adjustments. These are not phonological operations that occur in certain contexts, but they increase the overall pronunciation transparency of the romanized string. For example, appending a vowel to the vocalic letters as in Hindi "कृष्ण" (kr̥ṣṇa) "krishna" and Kannada "ಹೃದಯ" (hr̥daya) "hrudaya". Some clusters consistently have a different pronunciation, for instance "jñ" clusters are commonly pronounced as palatalized velars and are transcribed as "gy" as in Hindi "अज्ञान" (ajñāna) "agyaan". The Malayalam "rra virama rra" cluster denotes a geminated alveolar "t" as in "കുറ്റം" (kuṟṟaṁ) "kuttam". The same character sequence in Tamil is transliterated "tr" as in "குற்றம்" (kuṟṟam) "kutram".

Not all romanization operations are driven by pronunciation. Some natural transliteration conventions can introduce instability across scripts. For example, in some languages like Malayalam, Tamil and Kannada, the dental "t" is often transliterated as "th", as in Malayalam "താൾ" (tāḷˀ) "thal" and Tamil "தூங்க" (tūṅka) "thoonga". Thus "h" is used in these languages in this context to distinguish the dental from alveolar and retroflex "t". However, "h" is more typically used when romanizing other languages/scripts to indicate aspiration, so that "th" would typically only result from an aspirated dental. We introduce different output options in Section 4.2 that provide more similar output strings for similar input strings across different scripts.

Transliteration conventions for some of the long vowels can affect the pronunciation transparency negatively for readers who are unfamiliar with the conventions. While simply doubling the vowel to transliterate a long vowel is both common and closer to pronunciation, there are other conventions. A very common one is to simply use a single letter, making it ambiguous with the short vowel as in Hindi "फ़ारसी" "fārasī" "farsi". Another commonly seen convention is to transliterate a long "u" as "oo" or a long "i" as "ee", as in Tamil "தூங்க" (tūṅka) "thoonga" and Telugu "వీధి" (vīd^{h}i) "veedhi", respectively.

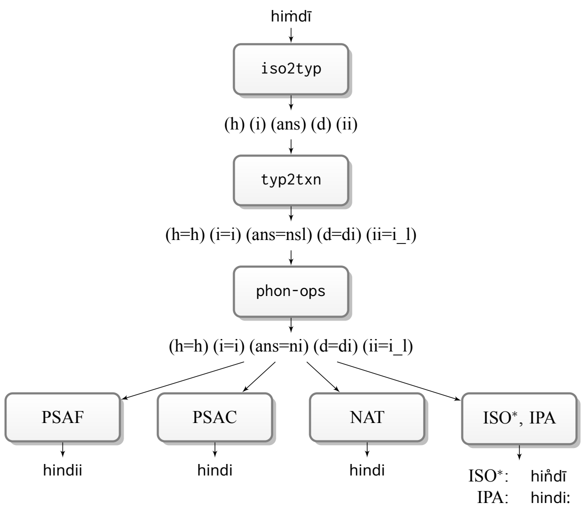

Figure 1: Diagram representing the main romanization components that operate on the input string "hiṁdī".

4.2. Implementation Details

In this section, we present specific implementation details of our approach, which maps from Brahmic script input to a number of possible outputs, via language/script-specific grammars written in either Thrax (Roark et al., 2012) or Pynini (Gorman, 2016; Gorman and Sproat, 2021) — the toolkits for representing and compiling regular expressions and context-dependent rewrite rules into finite state transducers (FSTs).[5] We present our approach as a series of stages, following the diagram in Figure 1.

4.2.1. Input Format

We make use of existing ISO 15919 romanization (ISO, 2001) so that the initial native script input to our approach is that romanized form. We assume that this corresponds to a string that has been visually-normalized to the canonical representation of visually identical strings. This reduces the number of alternatives that must be covered in the grammar.[6]

4.2.2. Intermediate Stages

Ease-of-input mapping: Since various grammars processing the input must be written — and for ease of interpretation — we first map this input romanization to an internal representation that uses only basic ASCII symbols (`iso2typ` in Figure 1).

Default phoneme mapping: The output of `iso2typ` is mapped to a default sequence of phonemes (`typ2txn` in Figure 1), following the unified phonological representation approach of Demirşahin et al. (2018). That paper focused on reducing the amount of training data required for text-to-speech systems by sharing data across South Asian languages, which required a phonological representation general enough to cover the 11 major Indo-Aryan and Dravidian languages (Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Nepali, Sinhala, Tamil, Telugu and Urdu). By using such a representation, we gain some traction on the issue of stability of romanization across languages, while still addressing pronunciation transparency.

Phonological operations: The default phoneme representation is then put through a number of common phonological operations (`phon-ops` in Figure 1), such as those discussed in Section 4.1. For example, schwa deletion is handled at this stage. We recast this as a general inherent vowel problem, which is dealt with via an insertion operation rather than deletion as is commonly used, i.e., we treat schwa as an epenthetic vowel (Hall, 2011).

Specifically, for schwa deleting languages we start by assuming all inherent vowels are silent, and insert only those that are required by the phonotactics. A possible inherent vowel is inserted if it is: (a) the first vowel of the word; (b) the last vowel before the last consonant(s) of the word; or (c) required as the nucleus of the syllable (before the coda cluster or after the onset cluster).

4.2.3. Output Types

After the phonological operations are applied, there are five different outputs that we can produce: natural transliteration (NAT); fine and coarse Pan-South-Asian romanization (PSAF and PSAC, respectively, in Figure 1); ISO-pronunciation (ISO*); and IPA. We discuss each of these in turn.

Natural transliteration: Natural transliteration is intended to be as close as to the natural user behaviour as possible. However there is no one true way to represent the user behaviour; not only because it is not standardised but also because it can be context and task dependent. Let's take the Hindi phrase "मैं वाटरलू गया" (maiṁ vāṭaralū gayā) in the context of a text message. If the user is romanizing the phrase to use a Latin keyboard as an input method with the expectation that the display text is going to be converted to Devanagari, they may choose to romanize it as "main vatarlu gaya" which matches the intended Devanagari characters somewhat closely. If the user intends the message to be displayed in Latin, maybe because the messaging app does not support Devanagari, the phrase can be romanized as "main waterloo gaya". This romanization may be perceived as more natural and it is more stable across scripts and languages, but due to the English orthography it loses some pronunciation transparency.

The natural transliteration output of our grammars aims to capture the user behaviour for text that is intended to be displayed in Latin, within the limitations of rule based grammars. Following the user behaviour we use only ASCII characters, which increases the ease of input but loses invertibility. Since it reflects

[5]Our full implementation in Pynini can be found at `https://github.com/google-research/nisaba/tree/main/nisaba/scripts/brahmic/natural_translit`.

[6]See Johny et al. (2021) for script transformations, NFC and beyond, that preserve visual invariance.

English Word	Hindi				Malayalam			
	Deva	ISO	PSAF	PSAC	Mlym	ISO	PSAF	PSAC
“Hindi”	हिंदी हिन्दी	hiṁdī hindī	hindii	hindi	ഹിന്ദി	hindi	hindi	hindi
“India”	इंडिया	iṁḍiyā	indiyaa	indiya	ഇന്ത്യ ഇൻഡ്യ	intya in’ḍya	indya	indya

Table 1: Native Devanagari and Malayalam strings and three corresponding romanization types: ISO-like, fine-grained pan-South Asian (PSAF) and coarse pan-South Asian (PSAC).

some of the phonological operations, such as nasal place assimilation and voicing, natural transliteration has more pronunciation transparency. However, we follow common romanization conventions such as different treatments of long vowels and gemination, which can negatively affect the pronunciation transparency and stability across scripts as discussed in Section 4.1. For example, the NAT transliteration for Malayalam “കേൾക്കാത്ത” (kēḷˁkkātta) is “kelkkaatha”. In this form, the first long vowel is transliterated as a single “e”, therefore introducing ambiguity for the vowel length, whereas the second vowel is transliterated as “aa”. The first gemination is transliterated as two letters, whereas the second one, marked as dental with an “h”, is not geminated in the transliteration. The same ISO substring “ātta” can have a completely different natural transliteration in another language, for example in Hindi “उदात्त” (udātta) “udatt”. The NAT format favours naturalness and conventionality over pronunciation transparency and stability across scripts by aligning with language and script specific user behaviour where it is contextually predictable.

Note that the recovery of an accurate English orthography is beyond the scope of rule based grammars, since it requires classification of the word as being English-origin.

Pan-South Asian transliteration: Pan-South Asian transliteration is a representation that favours consistency and stability above all else. There are two levels of this transliteration, shown in Table 1. The fine-grained level (PSAF) is a pronunciation-informed romanization of the token that ignores any language or region specific natural transliteration conventions. The PSAF output for For Malayalam “കേൾക്കാത്ത” (kēḷˁkkātta) is “keelkkaatta”, where both long vowels and both geminations are explicitly transliterated, and the language-specific dental marking is not applied. At this level, different spellings of the same word in one language are expected to have the same romanization, while the similarity of the romanization of the same word in different languages will increase compared to the ISO romanization. For example, in Hindi, the word “Hindi” has two spellings: “हिंदी” (hiṁdī) and “हिन्दी” (hindī), both have the same PSAF romanization “hindii”. The Malayalam counterpart “ഹിന്ദി” (hindi) has the PSAF romanization “hindi”. Similarly, the word “India” has two spellings in Malayalam: “ഇന്ത്യ” (intya) and “ഇൻഡ്യ” (in’ḍya), both PSAF romanized as “indya”, while the Hindi counterpart “इंडिया” (iṁḍiyā) is indiyaa.

The coarse-grained Pan-South Asian transliteration (PSAC) is deliberately under-specified in order to capture the similarity of tokens from different languages and scripts by choosing the shortest or the simplest form among the variations attested across languages and scripts, such as shortening long vowels or dropping geminations. For example, the word “Hindi” ends with a long vowel in Hindi “हिंदी” (hiṁdī) and a short vowel in Malayalam “ഹിന്ദി” (hindi). The PSAC romanization for both words is “hindi”.

ISO-pronunciation: The aim of this output form is to reflect the phonological operations that are relevant to the natural romanization on an ISO-like representation that preserves reversibility by using superscripts, subscripts, and diacritics. For example, the unpronounced inherent vowel in “फ़ारसी” “fārasī” “farsi’ in the ISO-pronunciation fār^{a}sī is represented as a superscript. The nasal in “हिंदी” (hiṁdī) “hindi” is assimilated to an “n” with a combining ring above that marks its origin as an anusvara in the ISO-pronunciation “hin̊dī” . If there are no phonetic operations like place assimilation, voicing, or schwa deletion, the output of ISO-pronunciation remains the same string as ISO.

ISO-pronunciation favours invertability over all else, while increasing pronunciation transparency. It is not a phonemic representation.

IPA: The IPA pronunciation (International Phonetic Association, 1999) output is not transliteration, but it is a conventional phonological transcription for linguistic readability. It is currently a direct representation of the phoneme sequence that results from applying the phonological operations that are related to romanization to the default multilingual phoneme mapping described above. It is not intended to be an exact transcription of the pronounced native word as is, since we currently do not apply any phonological rules other than those that are specifically relevant to the current task of romanization. For example phoneme sequences that are pronounced as diphthongs are left as vowel-vowel or vowel-consonant sequences in the IPA output since this operation has no impact on the transliteration.

4.3. Minimal Example in Pynini

This section presents a simplified implementation of the romanization pipeline using Pynini — a Python extension module for compiling, optimizing and applying finite-state grammars (Gorman, 2016; Gorman and Sproat, 2021). The grammars are compiled offline into collections of FSTs.

The initial stage of the romanization pipeline that maps the ISO input strings into internal ASCII representation (`iso2typ`) is implemented in Pynini grammar file `iso2typ.py` as follows:

```
iso2typ.py
import pynini as p
from pynini.export.multi_grm import (
  ExporterMapping as exporter)

iso_to_ascii = (
  (p.cross(”ā”, ”(aa)”) | ... # Vowels.
  | p.cross(”ṭ”, ”(tt)”) | ... # Consonants.
    # Symbols.
  | p.cross(”ṁ”, ”(ans)”) # anusvara.
  | p.cross(”'” : ”(chl)”) # chillu.
  | ...).star.optimize()

exporter[”ISO_TO_TYP”] = (
  iso_to_ascii @ p.cdrewrite(
    (p.cross(”(n)(chl)”, ”(n_chl)”) # Reconstruct.
    | ...), ””, ””, sigma_star).optimize()
```

The above snippet uses several core Pynini abstractions. We first define the `iso_to_ascii` FST that maps individual ISO letters representing vowels, consonants and auxiliary symbols to their corresponding ASCII counterparts using the FST union operator “`|`”. For example, this FST will map the individual ISO letter “ī” into the symbol corresponding to ASCII sequence “(ii)”. To transform this representation into an FST that operates on all possible sequences of the input ISO symbols, the Kleene star operator (Kuich and Salomaa, 1986), denoted “`star`”, is applied to `iso_to_ascii`. This FST is then composed (composition is represented in Pynini using “`@`”) with the context-dependent rewrite rule (defined using the `cdrewrite` function) that recombines the relevant letters and auxiliary symbols, as shown in the above snippet where the Malayalam *chillu* is combined with the corresponding nasal. In this particular example, we allow arbitrary left and right contexts for the rewrite rule defined over an alphabet of all possible ISO input symbols `sigma_star`. Finally, the resulting transducer is verified and possibly determinized and minimized using the Pynini `optimize` function and exported into a file.

The next stage `typ2txn` defines the mapping between the internal ASCII symbols and their corresponding default phonemic representation in the multilingual pan-South Asian phonology described above:

```
typ2txn.py
exporter[”TYP_TO_TXN”] = p.cdrewrite(
  p.cross(”(a)”, ”(a=a)”) # Vowels.
| ...
| p.cross(”(tt)”, ”(tt=tt)”) # Consonants.
| p.cross(”(t)”, ”(t=ti)”)
| ...
| p.cross(”(n_chl)” : ”(n_chl=ni)”)
| ...,
””, ””, sigma_star_typ).optimize()
```

The notable difference with the previous stage is that the context-dependent rewrite rule that performs the mapping operates over an alphabet `sigma_star_typ` that corresponds to the output symbols of `ISO_TO_TYP` FST from the previous stage.

A simplified snippet of the Pynini grammar that implements the phonological process of intervocalic voicing for languages like Malayalam as part of the larger phonological operations grammar `phon_ops.py` (phon-ops in Figure 1) is shown below:

```
phon_ops.py
letter = vowel_letter | consonant_letter
nasal = ”ni”
approximant = ”y”
sonorant = vowel_letter | nasal | approximant

exporter[”VOICING”] = p.cdrewrite(
  p.cross(”(tt=tt)”, ”(dd=dd)”),
  ”(” + letter.star + ”=” + sonorant + ”)”,
  ”(” + letter.star + ”=” + sonorant + ”)”,
  sigma_star_txn).optimize()
```

This context-dependent rewrite rule marks the unvoiced retroflex “tt” as voiced (“dd”) in the intervocalic position according to phonological process mentioned in Section 4.1.

An example Pynini grammar responsible for generating coarse and fine-grained PSA romanizations is shown below:

```
txn2nat.py
roman_coarse = p.cross(”a_l”, ”a”)
  | p.cross(”i_l”, ”i”) | ...
roman_fine = p.cross(”a_l”, ”aa”)
  | p.cross(”i_l”, ”ii”) | ...

exporter[”TXN_TO_PSAC”] = (p.cdrewrite(
  roman_coarse, ”=”, ”)”, sigma_star_phon
) @ remove_formatting).optimize()
exporter[”TXN_TO_PSAF”] = (p.cdrewrite(
  roman_fine, ”=”, ”)”, sigma_star_phon
) @ remove_formatting).optimize()
```

Both rewrite rules transform the phonemic element of each input symbol back to graphemic representation. The definition of the auxiliary context-dependent rewrite rule `remove_formatting` that removes helper decorations introduced by the intermediate stages is omitted for brevity.

The Pynini grammar snippet that shows all the component FSTs described above combined together in a single fine-grained PSA transducer (`ISO_TO_PSAF`) is shown below:

```
end2end_ml.py
import iso2typ as typ
import phon_ops as phn
```

Original word:	फ़ारसी	अस्पताल	വെളുത്ത
Language:	Hindi	Hindi	Malayalam
English gloss:	*Farsi*	*hospital*	*white*
ISO 15919:	fārasī	aspatāla	veḷutta
uconv:	farasi	aspatala	velutta
uroman:	phaarasii	aspataal	vellutta
NAT:	farsi	aspatal	velutha
PSAF:	faarsii	aspataal	velutta
PSAC:	farsi	aspatal	veluta
ISO-pron:	fār^{a}sī	aspatāl^{a}	veḷutta
IPA:	/fa:rsi:/	/əspəta:l/	/ʋeɭuta/

Table 2: Comparison of our various output types with existing romanization utilities *uroman* and *uconv* for a few example words.

```
import txn2nat as nat
import typ2txn as txn

exporter["ISO_TO_PSAF"] = (typ.ISO_TO_TYP
    @ txn.TYP_TO_TXN
    @ phn.VOICING
    @ nat.TXN_TO_PSAF).optimize()
```

The transducer is constructed by composing the relevant FSTs imported from four FST archives (FARs) `typ`, `phn`, `nat` and `txn` that represent the main stages of the romanization pipeline described earlier in this section (also see Figure 1). In the above example, the intervocalic voicing FST `VOICING` is applied prior to generating the romanization output and hence is suitable for languages like Malayalam.

Once the final transducers are compiled, the conversion between any ISO input and its corresponding fine-grained PSA representation can be achieved by composing an input string with the `ISO_TO_PSAF` transducer as shown by the following Pynini snippet for Malayalam:

```
input = "in'ḍya"
result = input @ ISO_TO_PSAF  # "indya"
```

At run-time, the resulting FSTs can be accessed from C++ using OpenFst (Allauzen et al., 2007) or from Python using Pynini libraries (Gorman, 2016; Gorman and Sproat, 2021).

4.4. Comparison with existing utilities

In this section, we compare the various outputs of our romanization approach on a few examples with those from other well-known romanization utilities, namely *uconv*[7] and *uroman*[8]. Amrhein and Sennrich (2020) compared these particular two romanization tools for model transfer in NMT. The *uconv* Unix command line tool provides a number of text mapping utilities including limited transliteration that is invertible in some cases. The *uroman* library (Hermjakob et al., 2018) provides non-invertible romanization for a large number of languages. Note that neither of these utilities is specifically designed for Brahmic script conversion of the sort investigated here, so this is in no way intended to promote our utility over them, rather to illustrate the differences that may arise in certain circumstances. Also, merely showing a few exemplars cannot substitute for an exhaustive validation; however, in this case, the comparison does provide some intuitions about the differences.

Table 2 presents two Hindi words and a Malayalam word, selected to illustrate how the various romanizations may differ or overlap. The Hindi word for Farsi (फ़ारसी) has long initial and final vowels, and, as noted in Section 4.1, the schwa in the 'ra' letter is deleted but not explicitly marked as such. Thus, the ISO 15919, uconv and uroman romanizations, which do not account for this schwa deletion, all have a vowel between 'r' and 's' despite it not being pronounced in the word. All of our outputs elide this vowel in the romanizations. Similarly, the final schwa in the second Hindi word (अस्पताल) is also deleted without an explicit virama, something that the uroman utility also detects in this case. The natural transliteration output (NAT) agrees with the coarse Pan-South-Asian (PSAC) output for both Hindi examples, but the Malayalam convention of romanizing dental 't' as 'th' (see Section 4.1) causes the NAT output to differ from PSAC in the Malayalam example. The fine-grained Pan-South-Asian (PSAF) output preserves gemination and vowel lengthening as does the uroman utility.

5. Conclusion and Future Work

We presented a number of criteria for useful romanization of South Asian languages, depending on the use scenario, and an approach to automatic romanization that attends to several of these criteria. The finite-state formalism used to provide the romanization utilities allows for inclusion of linguistic regularities as well as transliteration conventions for each language and script, so that the resulting romanizations can be tailored to the use scenario. This approach will be released in the Nisaba library at time of publication.

While several key criteria are addressed in the described approach, dealing with a number of common phenomena remain for future work. In particular, the criteria of conventionality is particularly important, and the significant subset of English-origin words (including acronyms) remain unaddressed in the currently described work.

6. Acknowledgements

Thanks to Raiomond Doctor, Anna Katanova, Lawrence Wolf-Sonkin and Christo Kirov for useful discussion around the issues presented in this paper, and to the anonymous reviewers for helpful suggestions.

[7]https://linux.die.net/man/1/uconv
[8]https://github.com/isi-nlp/uroman

Bibliographical References

Al-Onaizan, Y. and Knight, K. (2002). Machine transliteration of names in Arabic texts. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of 12th International Conference on Implementation and Application of Automata (CIAA)*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 11–23. Springer Verlag, July. Implementation and Application of Automata.

Amaladass, A. (2017). The writing catechism and translation strategies of three Jesuits in South India: Henrique Henriques, Roberto de Nobili and Joseph Beschi. In Antje Flüchter et al., editors, *Translating Catechisms, Translating Cultures*, volume 52 of *Studies in Christian Mission*, pages 170–194. Brill, Leiden, Netherlands.

Amrhein, C. and Sennrich, R. (2020). On Romanization for model transfer between scripts in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2461–2469, Online, November. Association for Computational Linguistics.

Annamalai, E. and Steever, S. B. (2019). Modern Tamil. In Sanford B. Steever, editor, *The Dravidian Languages*, Routledge Language Family Series, pages 118–175. Routledge, 2nd edition.

Antony, P. J. and Soman, K. P. (2011). Machine transliteration for Indian languages: A literature survey. *International Journal of Scientific & Engineering Research (IJSER)*, 2:1–8, December.

Appicharla, R., Gupta, K. K., Ekbal, A., and Bhattacharyya, P. (2021). IITP-MT at WAT2021: Indic-English multilingual neural machine translation using Romanized vocabulary. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*, pages 238–243, Online, August. Association for Computational Linguistics.

Arbabi, M., Fischthal, S. M., Cheng, V. C., and Bart, E. (1994). Algorithms for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–194, March.

Asher, R. and Kumari, T. C. (2012). *Malayalam*. Descriptive Grammars. Routledge.

Bhat, I. A., Mujadia, V., Tammewar, A., Bhat, R. A., and Shrivastava, M. (2014). IIIT-H system submission for FIRE2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation (FIRE'14)*, pages 48–53, Bangalore, India, December.

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

Cardenas, R., Lin, Y., Ji, H., and May, J. (2019). A grounded unsupervised universal part-of-speech tagger for low-resource languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2428–2439, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Chakrabortty, A. R. and Chakrabortty, B. (1976). On transcription and transliteration. *Annals of Library Science and Documentation*, 23(3):238–241, September.

Chakravarthi, B. R., Arcan, M., and McCrae, J. P. (2019). Comparison of different orthographies for machine translation of under-resourced Dravidian languages. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, pages 6:1–6:14, Leipzig, Germany, May.

Chakravarthi, B. R., Rajasekaran, N., Arcan, M., McGuinness, K., E. O'Connor, N., and McCrae, J. P. (2020). Bilingual lexicon induction across orthographically-distinct under-resourced Dravidian languages. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 57–69, Barcelona, Spain (Online), December. International Committee on Computational Linguistics (ICCL).

Chen, H.-H., Huang, S.-J., Ding, Y.-W., and Tsai, S.-C. (1998). Proper name translation in cross-language information retrieval. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 232–236, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July. Association for Computational Linguistics.

Datta, A., Ramabhadran, B., Emond, J., Kannan, A., and Roark, B. (2020). Language-agnostic multilingual modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8239–8243, Barcelona, Spain. IEEE.

Demirşahin, I., Jansche, M., and Gutkin, A. (2018). A unified phonological representation of South Asian languages for multilingual text-to-speech. In *Proceedings of the 6th International Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, pages 80–84, Gurugram, India, August.

Dutta, H. (2019). Assamese orthography: An introduction and some applications for literacy development. In R. Malatesha Joshi et al., editors, *Handbook of Literacy in Akshara Orthography*, volume 17 of *Literacy Studies (LITS)*, pages 181–194. Springer.

Flüchter, A. and Nardini, G. (2020). Threefold translation of the body of Christ: concepts of the Eucharist and the body translated in the early modern missionary context. *Humanities and Social Sciences Communications*, 7(1):1–16.

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.

Gutkin, A., Johny, C., Doctor, R., Wolf-Sonkin, L., and Roark, B. (2022). Extensions to Brahmic script processing within the Nisaba library: new scripts, languages and utilities. In *Proceedings of 13th Edition of Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Hall, N. (2011). Vowel epenthesis. In Marc van Oostendorp, et al., editors, *The Blackwell Companion to Phonology*, volume III. Phonological Processes, pages 1576–1596. John Wiley & Sons.

Hauer, B., Habibi, A. A., Luan, Y., Riyadh, R. R., and Kondrak, G. (2019). Cognate projection for low-resource inflection generation. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 6–11, Florence, Italy, August. Association for Computational Linguistics.

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.

Hermjakob, U., May, J., and Knight, K. (2018). Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia, July. Association for Computational Linguistics.

Huang, X., May, J., and Peng, N. (2019). What matters for neural cross-lingual named entity recognition: An empirical analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6395–6401, Hong Kong, China, November. Association for Computational Linguistics.

International Phonetic Association. (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, Cambridge, UK.

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.

Iyengar, A. (2015). Romanisation of Indian languages: a diachronic analysis of its failure. In *31st South Asian Language Analysis Roundtable Conference (SALA-31)*, Lancaster University, United Kingdom, May.

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.

Jones, A. (1975). The computer and material in Middle Eastern languages. *British Journal of Middle Eastern Studies*, 2(1):14–16.

Jung, S. Y., Hong, S., and Paek, E. (2000). An English to Korean transliteration model of extended Markov window. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

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.

Karimi, S., Scholer, F., and Turpin, A. (2011). Machine transliteration survey. *ACM Computing Surveys (CSUR)*, 43(3):1–46.

Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D. K., Aggarwal, P., Nagipogu, R. T., Dave, S., et al. (2021). MuRIL: Multilingual representations for Indian languages. *arXiv preprint arXiv:2103.10730*.

Khapra, M. M., Ramanathan, A., Kunchukuttan, A., Visweswariah, K., and Bhattacharyya, P. (2014). When transliteration met crowdsourcing : An empirical study of transliteration via crowdsourcing using efficient, non-redundant and fair quality control. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Khatri, J., Saini, N., and Bhattacharyya, P. (2021). Language relatedness and lexical closeness can help improve multilingual NMT: IITBombay@MultiIndicNMT WAT2021. In *Proceedings of* *the 8th Workshop on Asian Translation (WAT2021)*, pages 217–223, Online, August. Association for Computational Linguistics.

Knight, K. and Graehl, J. (1998). Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Krishnan, J., Anastasopoulos, A., Purohit, H., and Rangwala, H. (2021). Cross-lingual text classification of transliterated Hindi and Malayalam. *arXiv preprint arXiv:2108.13620*.

Kuich, W. and Salomaa, A. (1986). *Semirings, Automata, Languages*, volume 5 of *Monographs in Theoretical Computer Science*. Springer, Berlin.

Kunchukuttan, A., Puduppully, R., and Bhattacharyya, P. (2015). Brahmi-net: A transliteration and script conversion system for languages of the Indian subcontinent. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 81–85, Denver, Colorado, June. Association for Computational Linguistics.

Kunchukuttan, A., Jain, S., and Kejriwal, R. (2021). A large-scale evaluation of neural machine transliteration for indic languages. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3469–3475, Online, April. Association for Computational Linguistics.

Kunchukuttan, A. (2020). The IndicNLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library`.

Lee, J. S. and Choi, K.-S. (1998). English to Korean statistical transliteration for information retrieval. *Computer Processing of Oriental Languages*, 12(1):17–37.

Li, G. and Li, Y. (2019). Chinese Pinyin input method in smartphone era: A literature review study. In *Proceedings of 21st International Conference on Human-Computer Interaction (HCI)*, pages 34–43, Orlando, FL, USA, July. Springer.

Li, H., Zhang, M., and Su, J. (2004). A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 159–166, Barcelona, Spain, July.

Mahboob, T. S. and Rahman, M. M. A. (2017). Romanization in Bangladesh: Common malpractices. *Journal of Sociology, Nazmul Karim Study Center, University of Dhaka*, 9:7–23, June.

Mukhopadhyay, A., Dastidar, D. G., and Roy, M. K. (1985). A system for the machine processing of Bengali character strings. *IETE Journal of Research*, 31(3):84–88.

Muller, B., Anastasopoulos, A., Sagot, B., and Seddah, D. (2021). When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online, June. Association for Computational Linguistics.

Murikinati, N., Anastasopoulos, A., and Neubig, G. (2020). Transliteration for cross-lingual morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–197, Online, July. Association for Computational Linguistics.

Pandey, P. K. (1990). Hindi schwa deletion. *Lingua*, 82(4):277–311.

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

Patel, P., Mehta, M., Bhattacharya, P., and Atreya, A. (2020). Leveraging alignment and phonology for low-resource indic to English neural machine transliteration. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 373–378, Indian Institute of Technology Patna, Patna, India, December. NLP Association of India (NLPAI).

Pavan, K., Tandon, N., and Varma, V. (2010). Addressing challenges in automatic language identification of romanized text. In *Proceedings of 8th International Conference on Natural Language Processing (ICON-2010)*, Kharagpur, India, December.

Prabhakar, D. K. and Pal, S. (2018). Machine transliteration and transliterated text retrieval: a survey. *Sādhanā*, 43(6):1–25.

Rajan, V. (2020). Aksharamukha. `https://github.com/virtualvinodh/aksharamukha`.

Riyadh, R. R. and Kondrak, G. (2019). Joint approach to deromanization of code-mixed texts. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 26–34, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., and Tai, T. (2012). The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, Jeju Island, Korea, July. Association for Computational Linguistics.

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, Marseille, France, May. European Language Resources Association.

Ryskina, M., Gormley, M. R., and Berg-Kirkpatrick, T. (2020). Phonetic and visual priors for decipherment of informal Romanization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8308–8319, Online, July. Association for Computational Linguistics.

Ryskina, M., Hovy, E., Berg-Kirkpatrick, T., and Gormley, M. R. (2021). Comparative error analysis in neural and finite-state models for unsupervised character-level transduction. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 198–211, Online, August. Association for Computational Linguistics.

Sen, B. K. and Sur, S. N. (1979). Transliteration of Russian characters in English. *Annals of Library Science and Documentation*, 26(1-4):69–72.

Sinha, R. M. K. and Srinivasan, B. (1984). Machine transliteration from Roman to Devanagari and Devanagari to Roman. *IETE Journal of Research*, 30(6):243–245.

Veliath, C. (2011). Thomas Stephens — A human monument of inculturation in India. Bulletin of the Faculty of Foreign Studies 46, Sophia University, Tokyo, Japan.

Virga, P. and Khudanpur, S. (2003). Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, pages 57–64, Sapporo, Japan, July. Association for Computational Linguistics.

Wan, S. and Verspoor, C. M. (1998). Automatic English-Chinese name transliteration for development of multilingual resources. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*.

Wellisch, H. H. (1978). *The Conversion of Scripts: Its Nature, History, and Utilization*. Information sciences series. John Wiley & Sons, New York.

Wolf-Sonkin, L., Schogol, V., Roark, B., and Riley, M. (2019). Latin script keyboards for South Asian languages with finite-state normalization. In *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, pages 108–117, Dresden, Germany, September. Association for Computational Linguistics.

Zhang, Y., Wang, Z., Cao, R., Wei, B., Shan, W., Zhou, S., Reheman, A., Zhou, T., Zeng, X., Wang, L., Mu, Y., Zhang, J., Liu, X., Zhou, X., Li, Y., Li, B., Xiao, T., and Zhu, J. (2020). The NiuTrans machine translation systems for WMT20. In *Proceedings of the Fifth Conference on Machine Translation*, pages 338–345, Online, November. Association for Computational Linguistics.