A Rule-Based System for the Transcription of Sanskrit from the Devanagari Orthography to the International Phonetic Alphabet

Aalok Sathe
University of Richmond
28 Westhampton Way, Richmond, VA 23173, USA
aalok.sathe@richmond.edu

Abstract

We propose a new system for the transcription of Sanskrit text written using the Devanagari orthography, into the International Phonetic Alphabet, and supplement it with free and open-source software. We make use of existing literature on closest known pronunciations of sounds as well as prosodic and metric rules of syllabification using the Weerasinghe-Wasala-Gamage (WWG) algorithm for Sinhala, adapted to Sanskrit. We further incorporate suprasegmental sound changes along with the assignment of syllable-weight-determined stress.

Keywords: Transcription, Sanskrit, IPA, Phonetics, WWG algorithm, Devanagari, Computational linguistics

1. Introduction

The language Sanskrit is one of the oldest classical languages, and has a large amount of literature. For this reason, Sanskrit is a topic of frequent study in literature, culture, and linguistics. One hurdle in this process of studying is often the lack of an all-encompassing system of phonetic transcription. Whereas the IAST[1] and ITRANS[2] are widely used today, and will continue to be, they are but alternate means of representation of the same text, not fully capturing the phonological and prosodic features of the language. Additionally, it is our personal experience that even though systems such as IAST exist, students of Sanskrit worldwide have varying pronunciations of the same few sounds, seemingly approximated to the inventory of their primary languages. For new learners or even existing scholars, there can be a steep learning curve in Sanskrit phonology. Hence, it would be beneficial to new learners to have a system that would guide pronunciation as accurately and consistently as possible.

Some newer tools seemingly try to address this issue. However, they either do not solve the problem at hand, or do so inaccurately. Examples include the 'ICU' system for transliteration of Indic scripts (Viswanadha, 2002) as well as the website "Ashtangayoga" (Steiner, nd). In the case of the latter, we notice a lack of any indication of stress, syllabification, as well as that of within-word and suprasegmental phonological phenomena whatsoever. One may disregard these as 'superficial' details, but they are far from being that as syllabification and stress play an important role in classical Sanskrit poetic composition. We propose an improved system, hence, which we hope will serve as a convenient tool for reference in the study of Sanskrit phonology. We describe a system for the transcription of Sanskrit text written using its Devanagari orthography into the international phonetic alphabet (IPA). We choose IPA in particular to enable near-completeness of representation of the best-known pronunciations of Sanskrit sounds, rule-based syllabification adapted for Sanskrit from the Weerasinghe-Wasala-Gamage algorithm ('WWG algorithm') originally developed for the Sinhala language, and syllable stress: a prosodic feature not captured in any modern transcription system (e.g., IAST).

In this work, we aim to develop, based on existing work, a rule-based algorithmic system, and a computer program to supplement it, which will provide a consistent transcription given well-formed[3] Sanskrit text. We develop and distribute software accompanying this system, and license it using the GNU General Public License 3, to enable anyone to access and redistribute the source code as well as develop other software with the current implementation at its base. We believe this software will in itself be a tool for preservation of traditional knowledge as well as help create newer ones.

2. Sanskrit Phonology

Sanskrit is a classical language with its origins in the Indian subcontinent, and its literature and texts being found in present-day India, Nepal, and neighboring regions. Sanskrit is one of the official languages of India and shares close common ancestry with most of the modern Indo-Aryan languages spoken in the Indian subcontinent today (Emeneau, 1956) as well as some of the older Indo-European languages. It was recorded as the mother-tongue of about 14,000 people in the 2001 census of India (Banthia, 2001). While the effective pronunciations of Sanskrit sounds differ from region to region depending on the speaker's own mother tongue and regional linguistic influence, a unified approximation of Sanskrit sounds has been proposed in several existing works based on historical as well as present-day phonetic studies. In what follows, we attempt to give a summary of Sanskrit speech sounds.

In Sanskrit, there are multiple singular vowel sounds, as well as diphthongs made by combinations of individual vowels. The simple vowels are shown in table 1, and the diphthongs in 2. All of these vowels, whether simple or compound (diphthong), may be considered as a whole unit in Sanskrit for the purpose of prosodic analysis. Table 1 also shows the vowel length, which must accordingly be considered during transcription. Diphthongs are long vowels in Sanskrit. In addition, Sanskrit uses certain approximants and semivowels and treats them in the general category of vowels. These are

[1] International Alphabet of Sanskrit Transliteration

[2] Indian languages TRANSliteration

[3] That is, one adhering to the rules of classical Sanskrit phonology and Devanagari orthography

	Front	Central	Back
High	इ ([i]), ई ([iː])		उ ([u]), ऊ ([uː])
Mid	ϕ, ए ([eː])	अ ([ə]), ϕ	ϕ, ओ ([oː])
Low		ϕ, आ ([ɑː])	

Table 1: Sanskrit speech sounds: simple vowels. Symbols on the left are short variants of the vowel, while those on the right are long. In case a variant of a vowel does not exist, 'ϕ' is shown. Blanks denote vowels not in the Sanskrit phoneme inventory.

X	X+इ,ए	X+उ,ओ
अ,आ	ऐ ([ɑːi])	औ ([ɑːu])

Table 2: Sanskrit speech sounds: simplified rules of diphthong formation

ऋ ([ɻ̩]),	ॠ ([ɻ̩ː])
ऌ ([l̩]),	ॡ ([l̩ː])

Table 3: Sanskrit speech sounds: special vowels (sonorants). The first row shows short and long syllabic alveolar approximant sounds, respectively, while similarly, the second row shows short and long lateral approximant ones.

shown in table 3. For simplicity, we will consider all of these as vowels making up a single unit, just the way we do with "regular" vowels. Now, vowel length will be the only additional consideration other than identity, for the purposes of transcription.

We will base our transcription system upon existing literature on the phonology of Sanskrit (Jamison, 2004) as well as a system of correspondences between Devanagari text, IAST, and IPA, in the work 'The Original Pronunciation of Sanskrit' (Zieba and Stiehl, 2002). We will hence establish a mapping between Devanagari glyphs, their diacritic combinations if any, and IPA symbols (Association, 1999). Table 4 shows the correspondences used for consonants and other non-vowel sounds, while table 5 is the vowel and vowel-like sounds' counterpart. Sanskrit makes use of special symbols for several compound consonants, which we will process using their constituent components. Fortunately, Unicode character combinations for Devanagari define such compound characters in terms of their constituent components by default, making them easier to process. Although Sanskrit has many complex phonological processes where sounds interact (a popular one of which is *sandhi*), we need not encode rules of such phonetic interaction other than those implied by the orthography. This exclusion is because any phonological combination that occurs (such as from *sandhi*) results into a new phrase, which is written as-is in the orthography. It is expected of an input phrase to be well-formed, i.e., to not have any phonological inconsistencies per the rules of Sanskrit orthography. Given that this program will likely find use in transcription of existing texts, this should not be an issue in most cases.

We have taken the interpretation of sounds to be as close as possible to what is believed to have been the pronunciation in the classical Sanskrit era (Zieba and Stiehl, 2002; Jamison, 2004). One such noteworthy consideration is the differential pronunciations of a *visarga*, or the '◌ः'-terminal sound. Today, the interpretation of the pronunciation of this sound is slowly shifting towards a new trend: duplicating the vowel sound of the previous syllable after ([h]). For instance, रविः would end as [-ihi] according this rule as opposed to [-h]. While this seems to be a rising trend, it did not always use to be so, and the sound was supposed to be simply a [h]-terminating one, without vowel duplication.

3. Rule-Based Transcription

In our program, we will use several one-pass processes to fully transcribe a given text in linear time. In what follows, we describe some of the orthographic intricacies that require special attention in the design of the program.

3.1. Shorthand for Nasalization

The Devanagari Sanskrit orthography has several ways to indicate the presence of a nasal sound. Presence of nasals in a word is semantic, unlike some languages where it may have a conditioned occurrence. Nasals may be one of six types: five, derived from the conventional place of articulation (velar, palatal, retroflex, dental, and labial), and the sixth, simply a nasalized articulation of any vowel. Conventionally, a nasal consonant is only explicitly written when a phrase ends, or if the upcoming character is a vowel.[4] In case the nasal sound is not explicitly shown, an *anusvara* is shown on the character preceding it, and the actual sound corresponding to it is inferred from the forthcoming sound at the time of reading. For instance, if a word ends in a nasal sound, and the word after it begins with a bilabial stop, then the nasal is inferred to be [m]. When a sound does not belong to any of the five places of articulation mentioned above (e.g., a fricative, or a vowel), it shall be called the sixth case, and in this case, the preceding vowel is nasalized, with no additional sound being added. For instance, in the word संस्कृत ([sə̃s.kɻ̩.t̪ə]), where the *anusvara*'s circumstance is not one of the five types mentioned. The specific nasal sound to be used is inferred based on the next sound, if one exists, or is taken to be [m], the bilabial nasal sound, by default.

3.2. Handling the Default Schwa

A consonant character in Devanagari Sanskrit, unless explicitly marked *halant* (i.e., a schwa-less "partial" sound marked using the diacritic ◌्), has an implied schwa. For instance, ग may be transcribed as [gə], while to yield [g], we would need to mark a lack of schwa as ग्. Removal of schwa is required when either explicitly marking a character *halant*,

[4] As classified in the several tables above. A vowel in the strict phonetic sense is not meant here.

	Vl. plosive	Vl. aspirated plosive	Vd. plosive	Vd. aspirated plosive	Nasal	Approximant	Fricative
Glottal							ह [ɦə]**
Velar	क [kə]	ख [kʰə]	ग [gə]	घ [gʰə]	ङ [ŋə]		
Palatal	च [t͡ɕə]	छ [t͡ɕʰə]	ज [d͡ʑə]	झ [d͡ʑʰə]	ञ [ɲə]	य [jə]	श [ɕə]
Alveolar						र [ɹə] ल [lə]*	स [sə]
Retroflex	ट [ʈə]	ठ [ʈʰə]	ड [ɖə]	ढ [ɖʰə]	ण [ɳə]	ळ [ɭə]*	ष [ʂə]
Dental	त [t̪ə]	थ [t̪ʰə]	द [d̪ə]	ध [d̪ʰə]	न [nə]	व [ʋə]	
Labial	प [pə]	फ [pʰə]	ब [bə]	भ [bʰə]	म [mə]		

Table 4: Sanskrit speech sounds in Devanagari: consonants and non-vowel sounds. Merged cells indicate shared place of articulation. *Lateral approximants. **Voiced fricative.

Base	Diacritic	IPA	Base	Diacritic	IPA
अ		ə	आ	ा	ɑ:
इ	ि	i	ई	ी	i:
उ	ु	u	ऊ	ू	u:
ऋ	ृ	ɹ̩	ॠ	ॄ	ɹ̩:
ऌ	ॢ	l̩	ॡ	ॣ	l̩:
ए	े	e:	ऐ	ै	ɑ:i
ओ	ो	o:	औ	ौ	ɑ:u
अं	ं	əm	अः	ः	əh
ॐ		o:m			

Table 5: Sanskrit speech sounds: vowels and syllabic sounds.

or when combining it with another vowel, in which case, the vowel combination overrides the schwa. The way Devanagari diacritic combinations work in Unicode are from the point of view of typographic convenience. However, during transcription, we are required to explicitly remove the schwa, as demonstrated in the following example: गो = ग + ो is the way diacritic combination takes place in terms of Unicode characters. However, phonologically speaking, it is गो = ग + ् + ओ ([go:]), since we are removing the schwa and explicitly adding another vowel, instead of superficially dealing with diacritical marks. This needs to be taken care of during transcription, since, at the surface level, it is not explicit what underlying phonological process is taking place.

3.3. Syllabification

For syllabification, we implement the WWG algorithm (Weerasinghe et al., 2005) adapted to Sanskrit (Dasa, 2013). In the original study, the algorithm was developed to account for a majority of the Sinhalese vocabulary which has Sanskrit or Pali origins, as well as a large number of direct borrowings. In the same study, the authors note that the algorithm would be similarly applicable to Sanskrit with some modifications. As shown in algorithm 1, we use groups of vowel-consonant-vowel clusters (of the kind $V_B C_n C_{n-1} \ldots C_2 C_1 V_A$, where $n \geq 1$) for syllabification. Note that a cluster is not a syllable unit, but simply a device to locate syllable boundaries. We apply rules based on the number of consonants in the middle consonant cluster, i.e., n. Based on this length, prosodic syllabification conventions, we mark the boundaries of the syllables. We reuse boundary vowels, so a vowel that was processed while considering the current cluster will be included again to spot the next cluster. We achieve this by keeping track of indices where clusters began and ended.

Algorithm 1: WWG Algorithm adapted to Sanskrit

Input: Sanskrit text to be syllabified
`initialize` scope at the beginning of text;
while *end of text not reached* **do**
  `move` to next $V_B\mathbb{C}V_A$, where $\mathbb{C}$ is a consonant cluster;
  if *length of cluster* $\mathbb{C} = 1$ **then**
    `mark` syllable break after V_B;
  else if *length of cluster* $\mathbb{C} = 2$ **then**
    `mark` syllable break after first C from left;
  else if *length of cluster* $\mathbb{C} = 3$ **then**
    if *third consonant from left* = र् or य् **or** *first and second consonants are stops* **then**
      `mark` syllable break after first C from left;
    else
      `mark` syllable break before first C from right;
  else
    if *first consonant from right* = र् or य् **then**
      `mark` syllable break before second C from right;
    else
      `mark` syllable break after least sonorous C;
end
Result: Syllabified Sanskrit text

3.3.1. Examples

In what follows, we provide some example Sanskrit words to demonstrate syllabification as carried out using algorithm 1. For ease of reading, we highlight the consonant cluster in consideration using boldface in the Devanagari text.

1. कृ**तम्** ([kɹ̩.t̪əm]) was split before [t̪] following the rule for a cluster of length one.

2. व**ल्क**लानि (['ʋəl.kə.lɑ:.ni]): here, the first two syllables have been demarcated from each other by splitting a consonant cluster of length two.

3. (a) मत्स्यः ([ˈmət̪.sjəh]): this cluster of length three has been split according to the rule that checks the presence of either र् or य्.
   (b) उक्त्वा ([ˈuk.t̪ʊɑː]) demonstrates the rule involving two stops. Here, क् and त्. Hence, we split it after the first stop from the left hand side.
   (c) कृत्स्नम् ([ˈkɻt̪s.nəm]) is useful to illustrate the 'else' condition when the conditions similar to those in 3(a) and 3(b) do not apply.
4. कार्त्स्न्यम् ([kɑːɻt̪s.njəm]) contains a य्-terminal cluster of length more than three. We split it before the second consonant when scanning from the right.

3.4. Assigning Stress

Once we finish demarcating the syllables, we use traditional prosodic and metric rules to determine the syllables that should receive stress. In Sanskrit, a syllable is either 'light' (L) or 'heavy' (H) (Sridharan, 2005). A syllable may be considered to be light in the base case, which acquires the status of being heavy subject to meeting one or more of the following conditions.

1. Syllable contains a long vowel or diphthong
2. Syllable is nasal-terminated or has nasalized vowel
3. Syllable is stressed

The goal is to ensure that any syllable of the form $[C_{11}]V_1[C_{1n}...C_{13}]C_{12}[C_{21}...]V_2$ that results in a cluster of consonants because of the adjoining consonants of the next syllable (here, C_{21} and beyond), is heavy. If the syllable already satisfies at least one of the first two conditions above, it is already heavy. However, if not, we must use condition 3 and add stress to make it into a heavy one. For the sake of example, consider the syllables of the word कुरुक्षेत्र ([ku.ˈɻuk.ʂeː.t̪ɻə]). When taken independently, they have the weights L, L, H, L. However, when considered in the word, the character क्ष्, which is a compound consonant of क् + ष्, causes the previous non-heavy syllable (-[.ɻuk.]-) to end into a consonant cluster of consonants of adjoining syllables. It thus receive stress, and hence become heavy, making the weights of syllables L, H, H, L. The third syllable does not receive stress, even though the boundary of the syllable break after it, i.e., -[.t̪ɻə], contains a consonant cluster, due to having the long vowel े ([eː]), which satisfies the first condition.

4. Software

Prototype software developed as part of this work may be found at the following link: `https://github.com/aalok-sathe/sanskrit_IPA`. The program is written using Python3, primarily because of effortless inbuilt Unicode support. The program allows the user to transcribe text on-the-go using a command-prompt design. A command in the form: `transcribe` *text* may be used. The software can also read an input file externally and output it in a similarly named file. This may be especially useful for transcribing large texts. Specific implementations apart, the software has intuitively named methods and commented code that will allow anyone using it to build software on top. We observed a lack of permissively licensed software for this purpose, and would like to stress that the prototype program is free and open source software (FOSS) which may be used, modified, and redistributed by anybody in compliance with the GNU General Public License (version 3 or later). It is our hope that this licensing will encourage scrutiny, improvement, and further development in related research questions.

5. Future Work

We intend to evaluate the current work against hand-transcribed Sanskrit text. Evaluations will be hosted along with the source code. More work along similar lines will be required to create a set of tools to represent traditional knowledge in Sanskrit, as well as a large number of Indic languages. To begin with, systems need to be developed that will enable back-transcription from IPA to Devanagari, as well as all-way systems to transcribe consistently to most of the major ways of representing Sanskrit text today, such as ITRANS and IAST. Whereas developing such a system for Sanskrit is possible using rule-based decision procedures, it is not possible for most other modern Indic languages which rely largely on the speaker's cultural and experiential knowledge of the language for phonetic disambiguation. For such languages as Hindi and Marathi, statistical learning methods will need to be used in addition to rule-based systems to create transcription mechanisms that are accurate.

6. Acknowledgments

We are grateful for helpful comments by and discussion with Mukund Gokhale, Hema Kshirsagar, Dieter Gunkel, Shardul Chiplunkar, and Thomas Bonfiglio.

7. Bibliographical References

Association, I. P. (1999). *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. A Regents publication. Cambridge University Press.

Banthia, J. K. (2001). *Census of India, 2001*, volume 1. Controller of Publications.

Dasa, G. (2013). Sanskrit prosody: Syllabification with the WWG method. `sanskritstudio.wordpress.com/2013/09/13/`. Accessed: 2018-01-10.

Emeneau, M. B. (1956). India as a lingustic area. *Language*, 32(1):3–16.

Jamison, S. W. (2004). Sanskrit. *Cambridge Encyclopedia*, pages 673–699.

Sridharan, R. (2005). Sanskrit prosody, Pingala sutras and binary arithmetic. *Contributions to the History of Indian Mathematics, Hindustan Book Agency, Delhi*, pages 33–62.

Steiner, R. (n.d.). Transliteration tool. `https://www.ashtangayoga.info/sanskrit/transliteration/transliteration-tool/`. Accessed: 2018-01-10.

Viswanadha, R. (2002). Transliteration of Tamil and Other Indic Scripts. *Tamil Internet 2002*.

Weerasinghe, R., Wasala, A., and Gamage, K. (2005). A rule based syllabification algorithm for Sinhala. In *International Conference on Natural Language Processing*, pages 438–449. Springer.

Zieba, M. and Stiehl, U. (2002). The original pronunciation of Sanskrit. *Sanskritweb.net*.