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

Aalok Sathe
Department of Math & Computer Science
University of Richmond, VA, 23173
aalok.sathe@richmond.edu

Abstract

In this paper, we introduce a system for the transcription of Sanskrit text written using the Devanagari (or Nagari) script-based orthography for Sanskrit, into the International Phonetic Alphabet. We make use of existing literature on closest known approximate pronunciations of sounds as well as prosodic and metric rules for syllabification using the Weerasinghe-Wasala-Gamage algorithm for Sinhala adapted to Sanskrit and the assignment of syllable-weight-determined stress.

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

1. Introduction

The language Sanskrit, being one of the oldest classical languages in existence and having a huge amount of literature, is a topic of frequent study in literature, culture, and linguistics. One hurdle in this process often faced is the lack of a definite and consistent system of phonetic transcription. While the IAST[1] and ITRANS[2] have existed and are widely used, they are but alternate means of representation of the same text, not fully capturing the phonological and prosodic features of the text. For new learners or even existing scholars, there exists a steep learning curve as they learn Sanskrit phonology in the context of the language and culture. As such, it would be beneficial to new learners, existing scholars, and unrelated interested parties alike to have at their disposal such a system that would guide their pronunciation as accurately and consistently as possible. It is our personal experience that even after systems such as IAST existing for a while now, students and scholars of Sanskrit worldwide still fumble when it comes to phonology. Other systems have come into being too, however, they either do not solve the problem at hand, or do so inaccurately; examples (Viswanadha, 2002), (Steiner, n.d.). We propose a system, hence, which we believe will find use as a convenient reference tool as well as a tool for preservation of traditional knowledge by making it available to a wider audience. In this paper, we detail a system and a programmatic tool for the transcription of Sanskrit text written using its Devanagari orthography (hereafter referred to as simply “Devanagari Sanskrit”), into IPA, complete with the best-known pronunciations of its sounds, rule-based syllabification adapted for Sanskrit from the Weerasinghe-Wasala-Gamage algorithm (hereafter referred to as the “WWG algorithm”) originally developed for the Sinhala language, and most importantly, stress, a prosodic phonological feature not captured in such existing transcription systems.

In this work, we aim to provide a rule-based algorithmic system, and a computer program based on it, which will provide a consistent transcription given well-formed[3] Sanskrit text.

[1]International Alphabet of Saskrit Transliteration

[2]Indian languages TRANSliteration

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

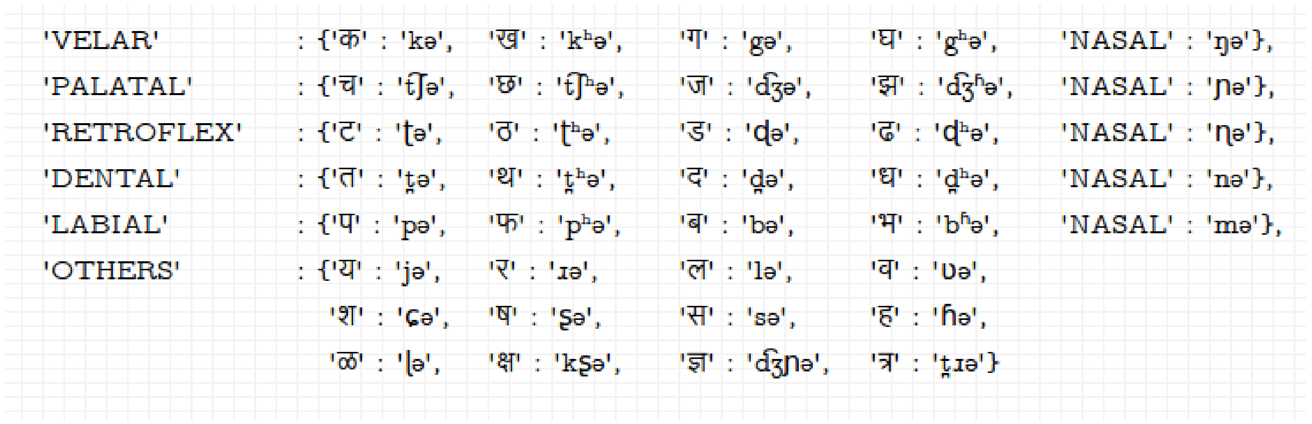
2. Sanskrit Phonology

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. It was recorded as the mother-tongue of 10,000 people in the 2001 census of India. While the effective pronunciations of Sanskrit sounds differ from region to region today depending on the speaker’s own mother tongue and such linguistic influence, a unified approximation of Sanskrit sounds has been proposed in multiple works. 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 vowels. The simple vowels are shown in table 1. The diphthongs made through the combination of two vowels, as shown in table 3. All these vowels, whether simple or compound (diphthong), are considered one whole unit in Sanskrit. Table 1 also shows the vowel length, which must accordingly be considered during transcription. Diphthongs are long vowels in Sanskrit. In addition, Sanskrit uses a few other special sounds and treats them as vowels. These are shown in table 2, and are various approximants. For the purposes of transcription, we will consider all these vowels to be a single unit each, with vowel length being the only additional consideration, in case two versions of the length of the same vowel are in existence.

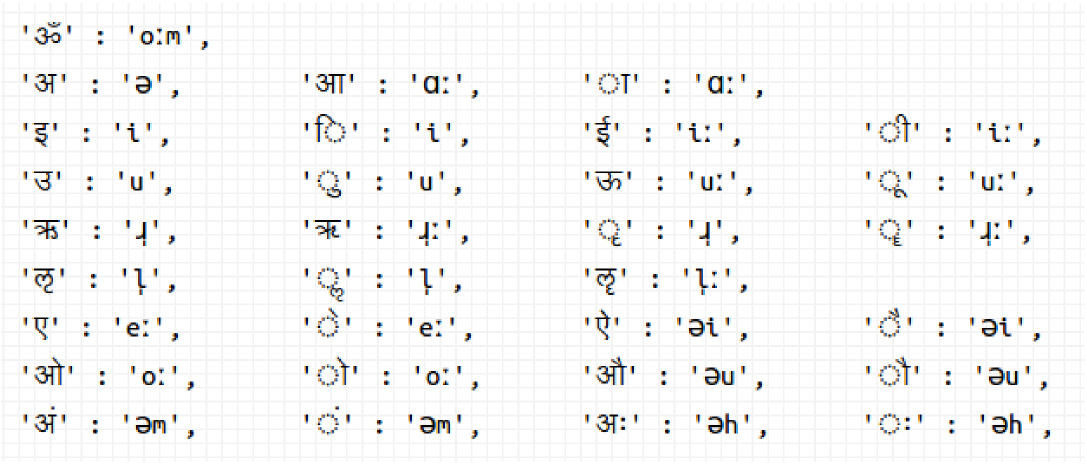
	Front	Central	Back
High	इ (/i/), ई (/iː/)		उ (/u/), ऊ (/uː/)
Mid	ϕ, ए (/eː/)	अ (/ə/), ϕ	ϕ, ओ (/oː/)
Low		ϕ, आ (/ɑː/)	

Table 1: Sanskrit speech sounds in Devanagari: simple vowels. Those toward the left are short versions of the vowel, while those toward the right are long. In case a short or long version of a vowel doesn’t exist, a ‘ϕ’ has been displayed to indicate the lack of it. Blank spaces denote that the vowel is not in the Sanskrit phonemic inventory.



```
'VELAR'      : {'क' : 'kə',   'ख' : 'kʰə',   'ग' : 'gə',    'घ' : 'gʰə',   'NASAL' : 'ŋə'},
'PALATAL'    : {'च' : 't͡ʃə',  'छ' : 't͡ʃʰə',  'ज' : 'd͡ʒə',   'झ' : 'd͡ʒʱə',  'NASAL' : 'ɲə'},
'RETROFLEX'  : {'ट' : 'ʈə',   'ठ' : 'ʈʰə',   'ड' : 'ɖə',    'ढ' : 'ɖʰə',   'NASAL' : 'ɳə'},
'DENTAL'     : {'त' : 't̪ə',   'थ' : 't̪ʰə',   'द' : 'd̪ə',    'ध' : 'd̪ʰə',   'NASAL' : 'nə'},
'LABIAL'     : {'प' : 'pə',   'फ' : 'pʰə',   'ब' : 'bə',    'भ' : 'bʱə',   'NASAL' : 'mə'},
'OTHERS'     : {'य' : 'jə',   'र' : 'ɹə',    'ल' : 'lə',    'व' : 'ʋə',
                'श' : 'ɕə',   'ष' : 'ʂə',    'स' : 'sə',    'ह' : 'ɦə',
                'ळ' : 'ɭə',   'क्ष' : 'kʂə',   'ज्ञ' : 'd͡ʒɲə',  'त्र' : 't̪ɹə'}
```

Figure 1: Sanskrit speech sounds in Devanagari: consonants and non-vowel sounds.



```
'ॐ' : 'oːm',
'अ' : 'ə',      'आ' : 'ɑː',     '◌ा' : 'ɑː',
'इ' : 'i',      'ि◌' : 'i',     'ई' : 'iː',     '◌ी' : 'iː',
'उ' : 'u',      '◌ु' : 'u',     'ऊ' : 'uː',     '◌ू' : 'uː',
'ऋ' : 'ɹ̩',      'ॠ' : 'ɹ̩ː',     '◌ृ' : 'ɹ̩',     '◌ॄ' : 'ɹ̩ː',
'ऌ' : 'l̩',      '◌ॢ' : 'l̩',     'ॡ' : 'l̩ː',
'ए' : 'eː',     '◌े' : 'eː',    'ऐ' : 'əi',     '◌ै' : 'əi',
'ओ' : 'oː',     '◌ो' : 'oː',    'औ' : 'əu',     '◌ौ' : 'əu',
'अं' : 'əm',     '◌ं' : 'əm',    'अः' : 'əh',     '◌ः' : 'əh',
```

Figure 2: Sanskrit speech sounds in Devanagari: vowels and syllabic sounds.

ऋ (/ɹ̩/),	ॠ (/ɹ̩ː/)
ऌ (/l̩/),	ॡ (/l̩ː/)

Table 2: Sanskrit speech sounds: special vowels. The first row shows short and long syllabic alveolar approximant sounds, respectively, while similarly, the second row shows short and long lateral approximant ones.

X	X+इ	X+उ
अ	ऐ (/əi/)	औ (/əu/)

Table 3: Sanskrit speech sounds: diphthongs

We will build upon an approximate pronunciation system that provides, after review, a transcription correspondence between Devanagari Sanskrit and IPA, in the work ‘The Original Pronunciation of Sanskrit’ (Zieba and Stiehl, 2002), to establish a mapping between Sanskrit and IPA symbols (Association, 1999). As such, figure 1 shows the correspondences used for consonants and other non-vowel sounds, while figure 2 is the vowel and vowel-like sounds’ counterpart. Sanskrit makes use of special symbols for several compound consonants as well, which we will process not as a whole, but using their constituent components. Fortunately, the unicode character combinations for Devanagari define these characters in terms of their constituent components, making it easier for us to process them. While 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 except those not captured explicitly in the orthography. It is expected of an input phrase to be well-formed, i.e., not having any phonological inconsistencies per the rules of Sanskrit. Given that this program will likely find use in transcription of existing texts, this should not be an issue in most cases. In case an input contains a character that is not present in the Devanagari Sanskrit orthography, the program should notice this, for it will make sure that the input characters are within a pre-specified unicode code-point range.

Another noteworthy consideration is the differential pronunciations of a *visarga*, or a ‘◌ः’-terminated character. Today, the interpretation of the pronunciation of this sound is slowly shifting towards a new one: sustaining the vowel sound of the previous syllable with an aspirated fricative (/h/) and duplicating the same vowel after the fricative sound. While this seems to be a rising trend, it did not always use to be so until very recently, and the sound was supposed to be simply a /h/-terminating one, without vowel duplication. As such, we have kept the interpretation of this sound as close as possible to what was believed to have emerged and in the classical Sanskrit era, and has remained the dominant accepted sound until recently (Zieba and Stiehl, 2002).

3. Rule-Based Transcription

In our program, we will use several one-pass processes to fully transcribe a given text, Computer Science-theoretically speaking, linearly complex in time. In what follows, some of the orthographic intricacies that require us to do so have been described.

3.1. Shorthand for Nasalization

The Devanagari Sanskrit orthography has several ways to indicate the presence of a nasal sound, which carries some meaning in the word's semantics. Conventionally, a nasal consonant is only explicitly shown when a phrase ends, or the upcoming character is a Sanskrit vowel[4]. In the case that the nasal sound is not explicitly shown, an *anuswaar* is shown on the character preceding it, and its actual sound is inferred at the time of reading or pronouncing. A nasal can be one of six types, five of which are shown in figure 1. The sixth type applies when a sound doesn't fit into any of the five categories, also shown in figure 1, which can be one of: velar, palatal, retroflex, dental, and labial, which are roughly the descriptions of sounds based on their sites of production. When a sound doesn't fall into any of these categories, for example, a fricative, or a vowel. In this sixth case, the preceding vowel is nasalized, and no additional sound is added. For instance in the word संस्कृत (/sə̃s.kɹ̩.t̪ə/) itself, where the *anuswar*'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*, has an implied schwa with it. For instance, ग may be transcribed as /gə/, while to yield /g/, we would need to mark a lack of schwa as ग्. As such, a removal of schwa may be required when explicitly marking a character *halant*, or when combining another vowel after it. In the latter case, it is the way Devanagari is structured in unicode code-points that requires this to be done: when combining a consonant with a non-schwa vowel, one must explicitly remove schwa first, as demonstrated in the following example: गो = ग + ो is the way the character combination takes place in unicode code-points, however, phonologically speaking, it is गो = ग + ् + ओ (/goː/), where we are removing the schwa and explicitly adding another vowel, instead of superficially dealing with diacritical marks. When dealing with the phonetic transcription, hence, this needs to be taken care of, since at the surface, it is not apparent what underlying phonological process is taking place.

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

3.3. Syllabification

For syllabification, we make use of the WWG algorithm (Weerasinghe et al., 2005) adapted to Sanskrit (Dasa, 2013). In the original study, the algorithm was developed to account for the majority of Sinhalese vocabulary which has Sanskrit and Pali origins as well as direct borrowings. As such, the authors note that the algorithm would be similarly applicable to Sanskrit. As shown in algorithm 1, we syllabify using groups of vowel-consonant-vowels, of the kind $V_B C_n C_{n-1} \ldots C_2 C_1 V_A$, where $n \geq 1$. We apply rules based on the number of such consonants in the central consonant cluster, i.e., n. Based on this length and conventional sonority- and meter-based syllabification preferences, we mark the boundaries of the syllables.

```
Input: Sanskrit text to be syllabified
initialize scope at the beginning of text;
while end of text not reached do
    move to next $V_B \mathbb{C} V_A$ group, 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;
        end
    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
    end
end
Result: Syllabified Sanskrit text
```

Algorithm 1: WWG Algorithm adapted to Sanskrit

3.3.1. Examples by Consonant Cluster Length

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**त**म् (/kɹ̩.t̪əm/) was split before /t̪/ following the rule for a cluster of length one.

2. व**ल्क**लानि (/ˈʋəl.kə.lɑː.ni/): here, the first two syllables have been demarcated from each other by splitting a consonant cluster of length two.

3. (a) म**त्स्य**ः (/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.t̪ʋɑː/) demonstrates the rule involving two stops, here, क् and त्. As such, we split it after the first stop from the left hand side.

(c) कृत्स्नम् (/ˈkɹ̩t̪s.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 (/kɑːɹt̪s.njəm/) contains a य्-terminal cluster of length more than three, and as such, it has been split before the second consonant when scanning from the right hand side.

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 or heavy (Sridharan, 2005). A syllable may be considered to be light in the base case, which later acquires the status of being heavy subject to meeting some conditions:

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

When a syllable ends in a consonant cluster instead of a single consonant (or none), the syllable receives stress to make it a heavy syllable if it is not already heavy. For the sake of an example, the syllables of the word कुरुक्षेत्र (/ku.ˈɹu.kʂeː.t̪ɹə/), when taken independently, are of the kind LLHL, with 'L' being light and 'H' being heavy. However, when considered together, the character क्ष, which is actually a compound consonant of क् + ष् + अ, causes the previous non-heavy syllable to receive stress, and hence become heavy, making the word of the kind LHHL. Interestingly, the third syllable does not receive stress, even though the syllable after it (i.e., the last one) contains a consonant cluster. This is because having the vowel sound ◌े has already caused it to become a heavy syllable.

4. Software

The software developed as part of this work may be found at the following link:
`https://github.com/aalok-sathe/sanskrit_IPA`.
The software, written using Python3, primarily because of inbuilt unicode support, allows the user to transcribe text on the go using a persistent command-prompt design, using a command in the manner: `transcribe` *text*. The software also has the option to set an input file externally, for large files, and output it in a similarly named file in the same directory. This is especially useful for transcribing large texts.
The software is released under the third version of the GNU General Public License, to support the spirit of free and open-source software, as well as to encourage further development in this regard.

5. Future Work

More work along similar lines will be required to truly create a set of tools to represent traditional knowledge in not just Sanskrit, but most of the Indic (of several language families) languages, and eventually, more underrepresented languages of the world. To begin with, systems need to be developed that will ensure backwards transcription compatibility of Devanagari Sanskrit and IPA, as well as those that will be able to transcribe consistently to most of the major ways of representing Sanskrit text today, including ITRANS and IAST. While developing such a system for Sanskrit is possible using rule-based decision procedures alone for the most part, this may not, and for a fact is not possible for most other modern languages, who 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 as accurate as possible.

6. Acknowledgments

We are grateful for helpful comments by and discussion on this and such related topics with Mukund Gokhale, Hema Kshirsagar, and Shardul Chiplunkar.
The fonts Annapurna SIL, Ubuntu, and Lohit Devanagari were used to render Devanagari text for the purpose of this manuscript. This manuscript was typeset using XeLaTeX.

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.

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

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.*