A Tool for Transliteration of Bilingual Texts Involving Sanskrit

Nikhil Chaturvedi
Indian Institute of Technology,
Hauz Khas, New Delhi, INDIA
cs5130291@cse.iitd.ac.in

Rahul Garg
Indian Institute of Technology,
Hauz Khas, New Delhi, INDIA
rahulgarg@cse.iitd.ac.in

Abstract

Sanskrit texts are increasingly being written in bilingual and trilingual formats, with Sanskrit paragraphs or shlokas followed by their corresponding English commentary. Sanskrit can also be written in many ways, including multiple ramanized encodings such as SLP-1, Velthuis etc. The need to handle code-switching in such texts is exacerbated due to the requirement of rendering web pages with multilingual Sanskrit content. These need to automatically detect whether a given text fragment is in Sanskrit, followed by the identification of the form/encoding, further selectively performing transliteration to a user specified script. The Brahmi-derived writing systems of Indian languages are mostly rather similar in structure, but have different letter shapes. These scripts are based on similar phonetic values which allows for easy transliteration. This correspondence forms the basis of the motivation behind deriving a uniform encoding schema that is based on the underlying phonetic value rather than the symbolic representation. The open-source tool developed by us performs this end-to-end detection and transliteration, and achieves an accuracy of 99.1% between SLP-1 and English on a Wikipedia corpus using simple machine learning techniques.

1 Introduction

Sanskrit is one of the most ancient languages in India and forms the basis of numerous Indian languages. It is the only known language which has a built-in scheme for pronunciation, word formation and grammar (Maheshwari, 2011). It is one of the most used languages of it's time and hence encompasses a rich tradition of poetry and drama as well as scientific, technical, philosophical and religious texts. Unfortunately, Sanskrit is now spoken by only a small number of people. The aforementioned literature, though available, remains inaccessible to most of the world. However, in recent years, Sanskrit has shown a resurgence through various media, with people reviving the language over the internet (Nair and Devi, 2011) and through bilingual and trilingual texts.

There exist numerous web-based application tools that provide age-old Sanskrit content to users and assist them with getting an insight into the language. Cologne Sanskrit Dictionary Project (Kapp and Malten, 1997) aims to digitize the major bilingual Sanskrit dictionaries. Sanskrit Reader Companion (Goyal and Huet, 2013) by INRIA has tools for declension, conjugation, Sandhi splitting and merging along with word stemming. Samsadhani (Kulkarni, 2017) by University of Hyderabad supports transliteration, morphological analysis and Sandhi. Sanskrit language processing tools developed at the Jawaharlal Nehru University (Jha, 2017) provide a number of tools with the final aim of constructing a Sanskrit-Hindi translator. In this paper, we attempt to construct a transliteration tool to render the web pages of the above tools in multiple scripts and encodings at the backend. Through this, we aim to expand the reach of Sanskrit to a wider community, along with the standardization of an open-source tool for transliteration.

The number of bilingual and trilingual content involving Sanskrit has been on a steady rise. For example, the Gita Supersite (Prabhakar, 2005) maintained by IIT Kanpur serves as a huge bilingual

database of the Bhagvad Gita, the Ramacharitmanas and Upanishads. Traditional texts such as Srisa Chandra Vasu's translation of the Ashtadhyayi in English (Vasu, 1897) exist in a similar format. These works broadly follow a commentary structure with Sanskrit hyms, verses and words followed by their translation in popular modern day languages like English or Hindi. Code-switching (Auer, 2013) is the practice of moving back and forth between two languages, or between two dialects/registers of the same language. Due to their commentarial nature, multilingual Sanskrit works constitute massive amounts of code-switching. For example, an excerpt of the Valmiki Ramayana from Gita Supersite: "तपस्वी ascetic, वाल्मीकिः Valmiki, तपः स्वाध्यायनिरतम् highly delighted in the practice of religious austerities and study of vedas, वाग्विदां वरम् eloquent among the knowledgeable, मुनिपुङ्गवम् preeminent among sages, नारदम् Narada, परिपप्रच्छ enquired." This motivates the need for a word-level transliteration tool that tackles areas of code-switching and performs transliteration through an automatic detection of the relevant sections.

Romanisation is another phenomenon that has led to the resurgence of Sanskrit on the Internet. In linguistics, romanisation is the conversion of writing from a different writing system to the Roman (Latin) script. Multiple methods of this transliteration have emerged, although none has emerged as the clear standard. These methods include SLP1, Velthuis, Harvard-Kyoto, ISO15919, WX, IAST and National Library at Kolkata romanisation. Such romanisation makes it easy for large parts of the world population to pronounce and appreciate Sanskrit verses. Therefore, any standardized transliteration tool for Sanskrit needs to support all the above romanisation encodings.

A property of the Sanskrit language and other major Indian languages like Hindi, Marathi, Tamil, Gujarati etc. that forms the basis of our transliteration, is that these languages are written using different letter shapes (scripts) but are rather similar structurally. The same sounds are duplicated across these languages, allowing for easy transliteration. The phonetic sound `[ki]` (IPA) will be rendered as कि in Devanagari, as ਕਿ in Gurmukhi, and as கி in Tamil. Each having different code- points in Unicode and ISCII[1]. This enabled us to formulate a mediating encoding schema that encodes the sound of a syllable rather than any syntactical aspect, thus allowing seamless transliteration between any 2 given scripts.

Romanised Sanskrit however exacerbates the problem of code-switching. The requirement for a general-purpose transliteration tool is now to differentiate between two words of the same script, which turns out to be a non-trivial problem. We again use the intuition of phonetics to overcome this problem. Certain sounds (or sequence of sounds) occur more frequently in some languages than in others. This allows us to formulate the classifier using a simple Naive Bayes model that functions on all possible substrings of a given word. We manage to achieve a classification accuracy of 99.1% between English and Sanskrit written using SLP1.

The rest of the paper is organized as follows. In section 2 we briefly describe presently used encoding and romanisation schemes for writing Sanskrit texts. Section 3 describes the prevalent transliteration tools available. Sections 4 and 5 respectively describe our transliterator and script detector. In section 6, we present our results and discuss possible future work.

2 Sanskrit Alphabets and Encodings

The Sanskrit alphabet comprises 5 short (ह्रस्व)[2] vowels, 8 long (दीर्घ) vowels and 9 prolated (प्लुत) vowels. Each of these vowels can be pronounced in three different ways: acute accent (उदात्त), grave accent (अनुदात्त) and circumflex (स्वरित). Vowels in acute accent are written as before (अ), in grave accent, a horizontal line is drawn under them (अ̱) and circumflex vowels are written with a vertical line drawn above

[1]Indian Script Code for Information Interchange (ISCII) is an 8-bit coding scheme for representing the main Indic scripts. Unicode is based on ISCII, and with Unicode being the standard now, ISCII has taken a back seat.

[2]In this paper, we use Unicode Devanagari enclosed within round brackets for better understanding through popular Sanskrit terms.

Table 1: Comparison of existing Transliteration Tools

Tool	Support for Encodings									Bilingual Support	Open Source
	Uni Dev	SLP1	ITR	Vel.	ISO	IAST	Harv. Kyoto	WX	Sohoni PE		
ITRANS	Yes	No	Yes	No	No	No	No	No	No	No	No
Sanscript	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes
Aksharamukha	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No
Samsaadhanii	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes
Google Input	Yes	No	No	No	No	No	No	No	No	No	No
Proposed Tool	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

them (अं). There are 33 consonants including 4 semi-vowels, 3 sibilants and 1 aspirate (ह).

There are several methods of transliteration from Devanagari to the Roman script (a process known as romanization) which share similarities, although no single system of transliteration has emerged as the standard. SLP1 (Scharf, 2011) and WX (Bharati et al., 1995) map each Devanagari letter to exactly one ASCII symbol. Velthuis (Velthuis, 1981) is based on using the ISO 646 repertoire to represent mnemonically the accents used in standard scholarly transliteration. IAST (Trevelyan et al., 1894) incorporates diacritics to represent letters. Harvard-Kyoto (Nakatani, 1996) largely resembles SLP1 in terms of using capital letters in its mapping. ITRANS (Chopde, 1991) exists as a pre-processing package and hence is widely used for electronic documents. ISO15919 (ISO/TC-46, 2001) like IAST uses diacritics. A comparison of some of the above schemes was first presented in (Huet, 2009). A more detailed comparison is also given under Appendix A of this paper. Reader may refer to (Scharf and Hyman, 2012) for a thorough analysis and discussion.

Unicode has designated code blocks for almost all major Indian scripts. The supported scripts are: Assamese, Bengali (Bangla), Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Oriya, Tamil, and Telugu among others. Across scripts, Unicode respects alphabet correspondence and letters with similar phonetic values are assigned the same code-points. As a result, transliteration can be done easily with a mere offsetting. In Unicode, the Devanagari symbol (अ) is coded as U+0905, whereas its representation in Gurmukhi script is (ਅ) which is coded as U+0A05. In comparison, the symbol (क) in Unicode Devanagari has its code as U+0915 while in Gurmukhi is (ਕ) with the code as U+0A15. Therefore, transliteration of Sanskrit texts written using Unicode in Indian scripts can be easily done by simply changing the offset value.

However, the Unicode encoding doesn't represent the language in its true essence. Hindi, Sanskrit and most other Indian languages are centred around phonetic values. Hence the encoded token should ideally represent the entire sound rather than it being split into different symbols for consonants and vowels. Since Unicode is based on the display of fonts and not the underlying phonetic structure, it requires significant parsing to figure out anything about the letter from its corresponding encoding, which section of consonants it belongs to, whether it is voiced or unvoiced etc. For example, the symbol (स्त्री) stands for the consonants (स्) and (त्) followed by the vowel (ई). Its Unicode representation will consist of (स + ◌् + त + ◌् + ई). Phonetically 3 units (स् and त् and ई), but represented in Unicode through 5 Unicode characters. Our tool fixes this issue by creating a new representation that encapsulates the consonants and the vowels (or lack of it) in a single encoding.

A comprehensive phonetic encoding (PE) based on the Ashtadhyayi rules for computational processing of Sanskrit language has been described in (Sohoni and Kulkarni, 2015). In order to implement this encoding in a general purpose programming language, a 20-bit encoding scheme was proposed. Although this encoding is phonetically rich, it is possible to compact it into fewer bits without compromising on the essential phonetic information present in the language. Our proposed internal encoding described in the following sections aims to achieve this goal.

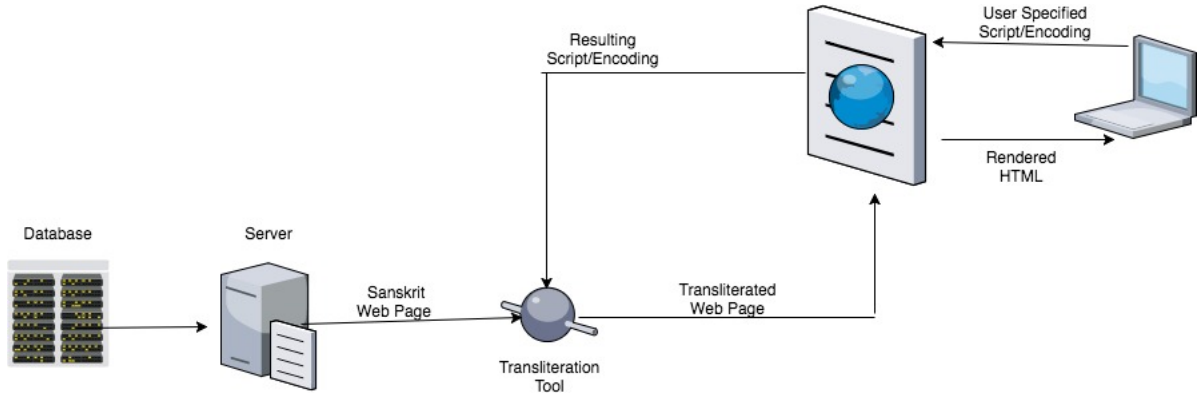

Figure 1: Model for Web-Based Applications

3 Existing Transliteration Tools

A number of tools exist as of today for Sanskrit transliteration to other scripts and encodings. We present a brief survey of the same. Aksharamukha (Rajan, 2001), Sanscript (Prasad, 2015) and ITRANS (Chopde, 1991) are some of the tools currently used for transliteration in Sanskrit. Google Input is another tool that is used to transliterating Devanagari to English. Though Aksharamukha and ITRANS support the romanised forms of Sanskrit, none of the aforementioned tools manage to handle bilingual scenarios. Most of these (except Sanscript) are also not open source and hence cannot be utilized by Sanskrit Developers. These tools have been summarised in Table 1.
International Phonetic Alphabet (IPA) is an internationally accepted scheme for encoding phonetic sounds. However, it has a number of representational and backward transliteration issues because of being completely sound based. The imported sounds (नुक्ता) don't share any correspondence to their roots. The sounds of (ऋ) and (रि) have the same representation in IPA, making it impossible to differentiate them while translating back. Anuswar (अनुस्वार) has multiple representations based on context, but none is unique to it (m, n, chandra). Visarga (विसर्ग) has the same representation as (ह).

WX and SLP encoding schemes are also phonetic in nature. However, the Sanksrit language alphabet system has a rich structure that categorizes the alphabets according to the place of pronounciation (Gutturals, Palatals, Retroflex, Dentals, Labials), the amount of air exhaled (aspirated or unaspirated) and whether the consonents are voiced and unvoiced. These attritutes of the alphabets are very useful while carrying out phonologial or morphological processing in the language. It is desirable to have an encoding that represents these attributes of the language in a natural manner.

Due to these inefficacies of existing tools and phonetic schemes, we created our own unified encoding schema which naturally encodes the sounds in the Sanskrit Varnamala (as described in the next section).

4 Design of the Transliterator

4.1 Internal Representation

We created an internal encoding that represents simple syllables (single consonant followed by single vowel) using 16-bits. Initial 5 bits in this encoding represent the script (hence can support 32 scripts). Next 6 bits represent the consonants (व्यंजन) while the last 5 bits represent the vowel (स्वर/मात्रा). Each 16-bit code represents a specific simple syllable sound, which can further be reverse mapped to a specified destination script. In contrast, the Unicode representation for a simple Sanskrit syllable would require 32-bits under the UTF-16 encoding, and 48-bits under the UTF-8 encoding.

With 33 consonants and 14 vowels, we can encode their permutations using just 9-bits versus the 11-bits

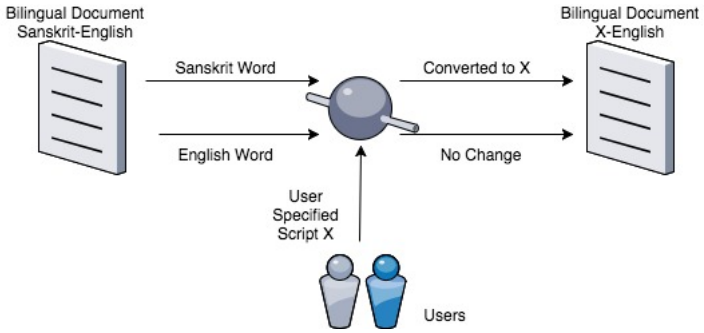

Figure 2: Model for Bilingual Texts

that we currently are using. But, we preferred to use some extra bits so as to keep our representation clean and allow for the bits within themselves to represent certain nuances of the Sanskrit language. Our encoding respects the phonetic structure of the language as described by Panini in a manner very similar to the phonetic encoding (PE) of (Sohoni and Kulkarni, 2015). Just by using the bit patterns of this encoding, it is possible to figure out important phonetic characteristics of the letters.

For the 5 bits of the vowels, the second-last bit represents whether the vowel is a simple vowel (अ, इ, उ, ऋ, ऌ) or a dipthong/compound vowel (ए, ऐ, ओ, औ). The last bit of the vowels represent the length of the vowel. Long (दीर्घ) vowels (आ, ई, ऊ, ॠ, ऍ, ए, ऐ, ओ, औ) will have their last bit as 1, while short (ह्रस्व) vowels (अ, इ, उ, ऋ, ऌ) will have their last bit as 0.

In the case of consonants, the first 3 bits represent the place of pronunciation of the letter. Thus, the sequence 000 refers to the throat as the source and the letters are called Gutturals (क, ख, ग, घ, ङ, ह), 001 refers to the palate and letters are called Palatals (च, छ, ज, झ, ञ, य, श). 010 refers to the murdha and are called Retroflex letters (ट, ठ, ड, ढ, ण, ऱ, ष), 011 contains letters with source of origin as the teeth and are called Dentals (त, थ, द, ध, न, ऌ, स). Lastly, 100 refers to the lips and the letters are called Labials (प, फ, ब, भ, म, व), while 101, 110 and 111 are reserved for special symbols and accents.

As for the last 3 bits of consonants, the first of these is 0 for stop-consonants (स्पर्श) which means non-nasal, non-semivowel and non-sibilant consonants. The second of these bits represents voicing (whether or not the vocal chords vibrate in pronunciation). It is 1 for voiced (घोष) consonants like (ग, घ) while 0 for unvoiced (अघोष) consonants like (क, ख). The last of these bits represents aspiration (a puff of air at the end of the pronunciation). It is 1 for aspirated (महाप्राण) consonants (ख, घ) while 0 for unaspirated (अल्पप्राण) consonants (क, ग). A table describing the proposed encoding is given in Appendix B.

4.2 Transliterator Pipeline

The transliterator takes a bilingual (or trilingual) document as its input and produces an output document in the same format where the Sanskrit text is transcribed into the specified script. It consists of 5 stages, namely fragmentation, script detection, tokenisation, universalisation and specification, explained below.

Fragmentation refers to splitting the given text into smaller fragments (words, sentences, paragraphs etc). The assumption shall be that the script and encoding remain same through these fragments if not through the entire text. In order to make it most general, currently fragmentation is done at the word level.

	Pred English	Pred SLP-1	Recall
Actual English	72294	1605	97.8%
Actual SLP-1	213	25004	99.2%
Precision	99.7%	93.9%	98.2%

Table 2: Confusion matrix of English vs Sanskrit-SLP-1 without proper noun correction

Script Detection refers to identification of the language, scripts and encodings for the various fragments through a Naive Bayes model described in section 5.

Tokenisation refers to splitting the fragment further into tokens, each of which represent a single sound. It is similar to the concept of English syllables. So the sound [ki] will be seen as one single token under this model.

Universalisation refers to the conversion of the token to the universal 16-bit encoding designed by us. This is done through pre-populated hash maps for different script tokens.

Specification refers to the conversion of the universal encoding to the specified script using pre-populated hash maps.

4.3 Use Cases

4.3.1 Web-based Applications

One of the foremost uses of our transliteration tool is it's utility for web-based applications. A number of websites nowadays serve the historical epics like the Gita and the Ramayana that were originally written in Sanskrit. Along with this, many websites also provide an avenue for people to learn Sanskrit grammar, understand conjugation and splitting of words, along with explaining the various forms of Sanskrit verb roots. Such websites are as of now available only in the Devanagari script. Our tool can be used to transliterate these pages to a user defined script/encoding at the backend. The model for this use case has been depicted in Figure 1. We insert our tool as a middle-ware between the backend and the frontend. The user specifies his required script/encoding on the frontend and all outgoing pages from the server pass through our tool while getting converted to that required script. The frontend then renders the converted HTML to the user for a seamless experience.

4.3.2 Bilingual Texts

Numerous Sanskrit texts have been modified to bilingual and trilingual texts through their translation to popular modern languages like English and Hindi. These works exist in a commentary form and incorporate massive amounts of code-switching. To represent any such text in a script different to that of its origin turns out to be an ordeal because the tool needs to conditionally perform the transliteration at a micro-level. This problem gets exacerbated when the Sanskrit verses are written using their Romanised form while the translation language is English. Figure 2 depicts the model for this use case.

4.3.3 User Driven

The third use for our tool is on the lines of Google input tools. Our tool can allow a user to enter a line of Sanskrit (in any script) intertwined with English and will output the resulting sentence to the user after transliteration. This not only provides an unmatched amount of flexibility to the user, but also has abundant relevance in the growing age of multi-lingual social media.

Basline 67.2%	Pred English	Pred SLP-1	Recall
Actual English	73178	721	99.0%
Actual SLP-1	205	25012	99.2%
Precision	99.7%	97.2%	99.1%

(a) English vs SLP-1

Basline 58.6%	Pred English	Pred Velthuis	Recall
Actual English	72649	1250	98.3%
Actual Velthuis	860	24357	96.6%
Precision	98.8%	95.1%	97.9%

(b) English vs Velthuis

Basline 51.1%	Pred English	Pred ITRANS	Recall
Actual English	72778	1121	98.5%
Actual ITRANS	645	24572	97.4%
Precision	99.1%	95.6%	98.2%

(c) English vs ITRANS

Basline 68.5%	Pred English	Pred HK	Recall
Actual English	73269	630	99.1%
Actual HK	199	25018	99.2%
Precision	99.7%	97.5%	99.2%

(d) English vs Harvard-Kyoto

Basline 73.4%	Pred English	Pred ISO	Recall
Actual English	73576	323	99.6%
Actual ISO	94	25123	99.6%
Precision	99.9%	98.7%	99.6%

(e) English vs ISO15919

Basline 71.5%	Pred English	Pred IAST	Recall
Actual English	73368	531	99.3%
Actual IAST	111	25106	99.6%
Precision	99.8%	97.9%	99.4%

(f) English vs IAST

Table 3: Confusion matrix of English vs Sanskrit using different Romanisation schemata

5 Design of the Script Detector

Differentiating English from Indian scripts, or differentiating different Indian scripts is easy as each uses a different alphabet with a different Unicode range. Hence, one can easily achieve a Word-level classifier with 100% accuracy. However, differentiating English text from Romanized Sanskrit/Hindi texts requires learning, specially to be able to do such classification at word-level. For this we designed a modified Naive Bayes classifier described next.

5.1 Modified Naive-Bayes Classifier

While learning, two dictionaries are maintained. The first dictionary compiles all seen complete words, while the other forms an occurrence database of all possible substrings of length <= 10. The intuition is that certain sounds (or sequence of sounds) occur more frequently in some languages then the others.

For a word, define the absolute frequency of a word as the actual number of occurrences for that word for a given language in the training dataset. On the other hand, the relative frequency of a given word is defined as its fraction of occurrences in the given language versus all other languages under consideration. While classifying, if the word is seen and the absolute as well as relative frequency is above a pre-set threshold for a particular language in training data, we classify it as that language. We use the relative frequency metric to account for mixed language nature of Wikipedia pages used as our dataset.

If the classifier encounters an unseen word, it is broken into all possible substrings of length >= 2 and

length <= 10. Subsequently, the probability of seeing a substring given a language, p(substr | lang), over all substrings of word using the trained substring dictionary is computed. This is a simplified version of the Naive Bayes model for the problem at hand. The word is classified to the language for which this metric turns out to be the maximum.

5.2 Training and Test Data

Training Data: One thousand random Wikipedia pages for both English and Sanskrit were used as the training data. The Sanskrit pages were converted to different Romanised Sanskrit encodings (such as SLP-1) using our universal encoder. We then parse out the irrelevant HTML meta-data and tags, stripping it down to just plain text content.

Test Data: One hundred more such random pages for both languages were used as the test data.

6 Results and Future Work

We tested our word-level language detection model on 100 random Sanskrit Wikipedia pages (after converting them to the 6 most popular romanisation schemes of SLP1, Velthuis, ITRANS, Harvard-Kyoto, ISO15919 and IAST). During our testing, we discovered that multiple English proper nouns like 'Bopanna' or 'Kannada' were getting classified as SLP-1 leading to a lower recall for English. In our opinion, such a misclassification aligns with the intention of the tool as it classifies the origin based on the prevalent sounds in the word. For Indian proper nouns appropriated to English these sounds still remain similar to those of their Sanskrit roots, and hence rather should be classified as that. These earlier results are presented in Table 2.

The final confusion matrices, obtained after manually removing proper nouns from the training and test dataset, are shown in Table 3. Each scheme shown has a corresponding baseline to compare our results with, shown in the top left cell. For SLP1, this baseline was the existence of a capital letter in the middle of a word. For Velthuis, it was the existence of a full stop in the middle of a word or the existence of doubly repeated vowels. For ITRANS, the baseline was similar to Velthuis, with repeated 'L' and 'R' instead of full stop. For Harvard-Kyoto, we selected the baseline as capital in the middle of the word alongside repeated 'L' and 'R'. Lastly, for ISO15919 and IAST, it was kept as the existence of a letter beyond the simple English letters and punctuation within a word.

As one can notice in Table 3, we in general attain a high precision for English and a high recall for the romanised words. A large number of misclassified words in both the English and SLP1 cases are 2-3 letter words. 'ati', 'ca' etc. are examples of SLP1 words misclassified as English, while 'Raj', 'Jan', 'are' etc. are examples of English words misclassified as SLP1. For these words, the modified Naive-Bayes model does not end up having enough information for correct classification.

We also tested our tool on a bilingual text test case by converting a extract from an English commentary on Ramayana from Gita-supersite (Prabhakar, 2005) to an mixture of SLP-1 and English. Subsequently, we converted the previous result back to Unicode Devanagari and English to see its differences with the original text. As can be seen in Figure 3, the transliteration from Devanagari-English to SLP1-English has a 100% accuracy due to our tool exploiting the difference in Unicode for the two scripts.

Our tool is available at `https://github.com/709nikhil/sanskrit-transliteration`. This tool can be further improved in several ways. The primary one being heuristically breaking down word into syllables rather than substrings, to provide a stronger basis for the phoneme intuition. One could also use machine learning approaches other than Naive Bayes, such as deep learning methods or conditional random fields (CRFs) (Lafferty et al., 2001). One could also incorporate contextual history into

तपस्वी ascetic, वाल्मीकिः Valmiki, तपः स्वाध्यायनिरतम् highly delighted in the practice of religious austerities and study of vedas, वाग्विदां वरम् eloquent among the knowledgeable, मुनिपुङ्गवम् preeminent among sages, नारदम् Narada, परिपप्रच्छ enquired.

(a) Original Bilingual Paragraph

tapasvI ascetic, vAlmIkiH Valmiki, tapaH svADyAyaniratam highly delighted in the practice of religious austerities and study of vedas, vAgvidAM varam eloquent among the knowledgeable, munipuNgavam preeminent among sages, nAradam Narada, paripapracCa enquired.

(b) Devanagari selectively transcribed to SLP1

तपस्वी ascetic, वाल्मीकिः Valmiki, तपः स्वाध्यायनिरतम् highly delighted in the practice of religious austerities and study of वेदस्, वाग्विदां वरम् eloquent among the knowledgeable, मुनिपुङ्गवम् preeminent among sages, नारदम् ङरद, परिपप्रच्छ enquired.

(c) SLP1-English transcribed back to Devanagari-English

Figure 3: Transliteration of Bilingual Texts

the transliteration to deal with the problem of incorrect classification of proper nouns, thereby aiming at a near perfect accuracy.

Acknowledgments

We thank the anonymous referees and the editors for their meticulous comments on the manuscript which helped in significantly improving the quality of the final paper.

References

Niraj Aswani and Robert J Gaizauskas. 2010. English-hindi transliteration using multiple similarity metrics. In 7th Language Resources and Evaluation Conference (LREC'10), pages 1786--1793, Valletta, Malta.

Peter Auer. 2013. Code-switching in conversation: Language, interaction and identity. Routledge, Daryaganj, Delhi, India.

Utsab Barman, Joachim Wagner, Grzegorz Chrupa□a, and Jennifer Foster. 2014. DCU-UVT: Word-level language classification with code-mixed data. In Proceedings of the First Workshop on Computational Approaches to Code Switching, pages 127--132.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific twitter collections. In Proceedings of the second workshop on language in social media (LSM'12), pages 65--74, Montreal, Canada.

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. Natural language processing: a Paninian perspective. Prentice-Hall of India, Delhi, India.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. Language Resources and Evaluation, 47(1):195--215.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), pages 161--175, Las Vegas, Nevada, USA.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using CRF: Code-switching shared task report of MSR India system. In Proceedings of The First Workshop on Computational Approaches to Code Switching, pages 73--79, Doha, Qatar.

Avinash Chopde. 1991. Indian languages transliteration (itrans). https://www.aczoom.com/itrans/.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a Sanskrit reader. In Proceedings of 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd, pages 130--171, IIT Bombay, India.

Pawan Goyal, Gérard P Huet, Amba P Kulkarni, Peter M Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In Proceedings of Conference on Computational Linguistics (COLING'12): Technical Papers, pages 1011--1028, Mumbai, India.

Gérard Huet, Amba Kulkarni, and Peter Scharf. 2009. Sanskrit computational linguistics 1 & 2. LNAI 5402, Springer-Verlag.

Gérard Huet. 2009. Formal structure of Sanskrit text: Requirements analysis for a mechanical Sanskrit processor. In Proceedings of 3rd International Symposium on Sanskrit Computational Linguistics (LNAI, vol 5402), pages 162--199, University of Hyderabad, India.

ISO/TC-46. 2001. ISO 15919 - Transliteration of Devanagari and related Indic scripts into Latin characters. https://www.iso.org/standard/28333.html.

G. N. Jha. 2017. Sanskrit Sandhi recognizer and analyzer. http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp.

Dieter B Kapp and Thomas Malten. 1997. Report on the Cologne Sanskrit dictionary project. Read at 10th International Sanskrit Conference, Bangalore.

Amba Kulkarni. 2017. Samsadhani: A Sanskrit computational toolkit. http://sanskrit.uohyd.ac.in/scl/.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the 18th International Conference on Machine Learning (ICML'01), pages 282--289, Williamstown, MA, USA.

Krishna Maheshwari. 2011. Features of Sanskrit. Hindupedia.

R Raman Nair and L Sulochana Devi. 2011. Sanskrit Informatics: Informatics for Sanskrit studies and research. Centre for Informatics Research an Development.

H. Nakatani. 1996. Harvard Kyoto. https://en.wikipedia.org/wiki/Harvard-Kyoto.

Dong Nguyen and A Seza Do□ruöz. 2013. Word level language identification in online multilingual communication. In Proceedings of the 2013 conference on Empirical methods in Natural Language Processing (EMNLP'13), pages 857--862, Seattle, Washington, USA.

Peeta Basa Pati and A. G. Ramakrishnan. 2008. Word level multi-script identification. Pattern Recognition Letters, 29(9):1218--1229.

Sheldon Pollock. 2006. The language of the gods in the world of men: Sanskrit, culture, and power in premodern India. University of California Press, Berkeley, California, USA.

T.V. Prabhakar. 2005. Gita supersite : Repository of Indian philosophical texts. https://www.gitasupersite.iitk.ac.in.

V. K. Prasad. 2015. Sanscript. http://www.learnsanskrit.org/tools/sanscript.

Vinodh Rajan. 2001. Aksharmukha. http://www.virtualvinodh.com/wp/aksharamukha/.

Peter M Scharf and Malcolm Donald Hyman. 2012. Linguistic issues in encoding Sanskrit. Motilal Banarsidass Publishers, Kamla Nagar, New Delhi, India.

Peter Scharf. 2011. Sanskrit library phonetic basic encoding scheme (SLP1). https://en.wikipedia.org/wiki/SLP1.

Samir Sohoni and Malhar Kulkarni. 2015. Character encoding for Computational Ashtadhyayi. In Proceedings of 16th World Sanskrit Conference (WSC'15): Sanskrit and the IT world, Bangkok.

Charles Trevelyan, William Jones, and Monier Williams. 1894. International alphabet of Sanskrit transliteration. https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration.

Srisa Chandra Vasu. 1897. The Ashtadhyayi of Panini. Sahitya Akademi, Rabindra Nagar, New Delhi, India.

Frans Velthuis. 1981. Velthuis. https://en.wikipedia.org/wiki/Velthuis.

Appendix A: Comparison of various Devanagari Romanisations

Devanagari	Unicode	Velthius	SLP-1	WX	ITRANS	Harvard-Kyoto	IAST	ISO-15919
अ	U+0905	a	a	a	a	a	a	a
आ	U+0906	aa	A	A	A/aa	A	ā	ā
इ	U+0907	i	i	i	i	i	i	i
ई	U+0908	ii	I	I	I/ii	I	ī	ī
उ	U+0909	u	u	u	u	u	u	u
ऊ	U+090A	uu	U	U	U/uu	U	ū	ū
ए	U+090F	e	e	e	e	e	e	ē
ऐ	U+0910	ai	E	E	ai	ai	ai	ai
ओ	U+0913	o	o	o	o	o	o	ō
औ	U+0914	au	O	O	au	au	au	au
ऋ	U+090B	.r	f	q	RRi/Rî	R	ṛ	r̥
ॠ	U+0960	.rr	F	Q	RRI/RÎ	RR	ṝ	r̥̄
ऌ	U+090C	.l	x	L	LLi/Lî	lR	ḷ	l̥
ॡ	U+0961	.ll	X		LLI/LÎ	lRR	ḹ	l̥̄
अं	U+0902	.m	M	M	M/.n/.m	M	ṃ	ṁ
अः	U+0903	.h	H	H	H	H	ḥ	ḥ
अँ	U+0904		~	z	.N			m̐
ऽ	U+093D	.a	'	'	.a	'	'	'
क	U+0915	ka	ka	ka	ka	ka	ka	ka
ख	U+0916	kha	Ka	Ka	kha	kha	kha	kha
ग	U+0917	ga	ga	ga	ga	ga	ga	ga
घ	U+0918	gha	Ga	Ga	gha	gha	gha	gha
ङ	U+0919	"na	Na	fa	Na	Ga	ṅa	ṅa
च	U+091A	ca	ca	ca	cha	ca	ca	ca
छ	U+091B	cha	Ca	Ca	Cha	cha	cha	cha
ज	U+091C	ja	ja	ja	ja	ja	ja	ja
झ	U+091D	jha	Ja	Ja	jha	jha	jha	jha
ञ	U+091E	na	Ya	Fa	na	Ja	ña	ña
ट	U+091F	.ta	wa	ta	Ta	Ta	ṭa	ṭa
ठ	U+0920	.tha	Wa	Ta	Tha	Tha	ṭha	ṭha
ड	U+0921	.da	qa	da	Da	Da	ḍa	ḍa
ढ	U+0922	.dha	Qa	Da	Dha	Dha	ḍha	ḍha
ण	U+0923	.na	Ra	Na	Na	Na	ṇa	ṇa
त	U+0924	ta	ta	wa	ta	ta	ta	Ta
थ	U+0925	tha	Ta	Wa	tha	tha	tha	Tha
द	U+0926	da	da	xa	da	da	da	Da
ध	U+0927	dha	Da	Xa	dha	dha	dha	Dha
न	U+0928	na	na	na	na	na	na	na
प	U+092A	pa	pa	pa	pa	pa	pa	pa
फ	U+092B	pha	Pa	Pa	pha	pha	pha	pha
ब	U+092C	ba	ba	ba	ba	ba	ba	ba
भ	U+092D	bha	Ba	Ba	bha	bha	bha	bha
म	U+092E	ma	ma	ma	ma	ma	ma	ma
य	U+092F	ya	ya	ya	ya	ya	ya	ya
र	U+0930	ra	ra	ra	ra	ra	ra	ra
ल	U+0932	la	la	la	la	la	la	la
व	U+0935	va	va	va	va/wa	va	va	va
श	U+0936	"sa	Sa	Sa	sha	za	śa	śa
ष	U+0937	.sa	za	Ra	Sha	Sa	ṣa	ṣa
स	U+0938	sa	sa	sa	sa	sa	sa	sa
ह	U+0939	ha	ha	ha	ha	ha	ha	Ha

Appendix B: Proposed Encoding Schema

The characters in our encoding schema are represented in 16 bits as follows: $b_{15}...b_{11}$-$b_{10}b_9b_8$-$b_7b_6b_5$-$b_4b_3b_2$-b_1b_0. The bits $b_{15} - b_{11}$ are used to represent the script. The remainder bits are represented as given in the tables 4 and 5 below. Null for the consonant part represents a pure vowel, whereas null for the vowel part represents consonants without a vowel sound. For example, अ is represented as 00000-111-111-000-00. The symbol क्ष which is broken up as (क् + ष = क् + ष् + अ) will be represented as two 16-bit units, 00000-000-000-111-11 representing (क्) and 00000-010-101-000-00 representing (ष).

$b_{10}b_9b_8$ \ $b_7b_6b_5$	000	001	010	011	100	101	110	111
000	क्	ख्	ग्	घ्		ह्	ङ्	
001	च्	छ्	ज्	झ्		श्	ञ्	य्
010	ट्	ठ्	ड्	ढ्		ष्	ण्	र्
011	त्	थ्	द्	ध्		स्	न्	ल्
100	प्	फ्	ब्	भ्			म्	व्
101	क़्	ख़्	ग़्	ज़्	ड़्	ढ़्	फ़्	
110	Udatta	Anudatta	ः	ं	ँ	ऽ		
111	Latin	Punc.	Num.	Vaid.				Null

Table 4: Mapping for 6 consonant bits

$b_4b_3b_2$ \ b_1b_0	00	01	10	11
000	अ	आ		
001	इ	ई		ए
010	ऋ	ॠ		ऐ
011	ऌ	ॡ		ओ
100	उ	ऊ		औ
101				
110				
111				Null

Table 5: Mapping for 5 vowel bits