International Journal of Recent Engineering Research and Development (IJRERD)
ISSN: 2455-8761
www.ijrerd.com || Volume 03 – Issue 10 || October 2018 || PP. 19-23

Transliteration of Kannada Text to English Text

Kuche Anurag
Junior Manager at India Tech

Kuche Bhavani Priya
Dept of Information Science and Engineering, RVCE

Karthik Kashyap
Dept of Information Science and Engineering, RVCE

Abstract: Transliteration system is one of the important Challenge in Natural Language Processing (NLP) domain. This system is all about converting one language script into another language script based on the International Phonetic Alphabet (IPA) rules. Transliteration is a very important NLP tool required for translating. Many languages like Indian, Arabic, etc. used the Roman script to represent their original language script. The major difficulty in transliteration is predicting the pronounce of the original word.
In this paper, we analyze the method used for transliteration of Kannada text to English text. This paper is intended to give a brief description of the method we used for transliteration of Kannada text to English text.
Keywords: Unicode, International Phonetic Alphabet(IPA), Natural Language Processing(NLP)

Introduction

Language is all about developing, maintaining and complex system of communication. Its either written or spoken and used by particular country or community. A script is a form of a written document where people use it to have a record of what our ancestors used for communication. Transliteration is the practice of translating, putting or writing a character or word from one alphabetical system into another alphabetical system. The model must be designed in such a way that the phonetic structure of words should be preserved and the meaning of the words must be kept as close as possible.

In translation, letters from one language are mapped to letters in a different language. The script used for obtaining these texts is not given a lot of importance. The source language text, as well as the target language text, can be in any suitable Script. Considerable knowledge of the language is required to structure this model. Every alphabet has their own Unicode. The transliteration also depends on the context. To transliterate names, we have to check the phonetic correspondence of alphabets and substrings in Kannada to English. For example, 'f' and 'ph' both map to the same sound 'f'. There is no much difference between the pronunciation of "i" and "ai". So it depends on the context of the text which is to be used.

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter language. Like vowels and consonants, there are even other divisions in alphabets based on the way they are pronounced and the organ used for pronunciation. The diphthong is a class of the division in vowels. Diphthongs are also called as the gliding vowel and are a combination of two adjacent vowel sounds. The tongue moves during the pronunciation of the diphthongs.

Same as vowels even consonants are divided into eight divisions based on the way they are pronounced and the organ used for pronunciation. They are:

1) Gutturals – Ka, Kha, ṅ etc
2) Palatals – ca, cha, ñ etc
3) Cerebral – ṭa, ṭha, ṇ etc
4) Dentals – ta, tha, n etc
5) Labials – pa, pha, m etc
6) Semivowels – ya, ra etc
7) Sibilants – śa,ṣa etc
8) Aspirants – ha

1. The vowels that are at the beginning of the syllable are considered. Consonants followed by the vowels can be found in grammars. No vowels and consonants have a difference in transliteration.

2. Vowel "a" is followed by almost any of the consonants so we need to consider few exceptions:
a) when any other vowel is indicated by its appropriate sign and

b) when the absence of any vowel is indicated by the superscript sign (◌◌ೆ).

3. Exception: Anusvara is transliterated by:

a) ṅ before gutturals,
b) ñ before palatals,
c) ṇ before cerebrals,
d) n before dentals, and
e) m before labials.

Why Romanization of kannada is necessary?

We write texts in Indian languages using the Roman alphabet. This process is known as Romanization. This can be bidirectional and if so, we can actually map from any script to any other script via Roman. The idea presented in this paper are generic and applicable to other language and script scenarios anywhere in the world.

ಸಸಹ ಮತತತಮತದ ಮಲ. (1)

To translate this to the romanized English script, we need to first split the words into single root characters. Each root character will have their own specific Unicode which can be obtained by using hex(ord(char)) function.

When we try to split the first word of the statement(1), the number of Unicode obtained might be more than the characters displayed. Such as ಸಸಹ - ಸ+◌ಿಿ+ ◌ಿಸ +ಹ will be split in this way. When translated to English script it returns sa+i+M+ha making the word saiMha. But the target word is siMha.

In order to simplify the characters of Kannada language we make use of python dictionaries. We create two dictionaries in which one contains all the consonants and other contains vowels. Consonants dictionary contains numerical and special symbols. These dictionaries do not contain the Kannada character themselves instead they Unicode values of Kannada characters in string form as keys. These keys will have their respective English character in string form as their values.

With the help of these dictionaries, we are able to transform their Kannada characters into their English characters. When a Kannada character is taken as input their respective Unicode value is obtained and is matched with one of the keys in the dictionary. If there is a match then their value will be returned as the output, if not then the loop terminates.

If the Unicode of the current character is present in consonants and if the next character Unicode is not present in vowels then the key of the current Unicode is written to the file. If the Unicode of the current character is present in consonants and if the next character Unicode is present in vowels then the last letter of the key element of the current character Unicode is removed and appended with the key value of the next character Unicode.

Even the special symbols like Full-stop(.), Comma(,)etc have their respective Unicode as that also has to be printed in the target file. When we encounter hex(ord(char))= ‘0xccd' which is ◌ಿೆ we just continue the code as usual. Because this pronunciation matters but is not considered in written text.

Advantages
1) To obtain English script of Kannada language using suitable rendering algorithm.
2) It can help readers who do not know Kannada script but understands Kannada.
3) Processing of texts rendered in different scripts may require different techniques for dealing. It can be simple, direct, natural and efficient.

For some letters in Kannada which needs some effort in pronunciation, for example, Kha, M, Tta, etc. We use h to indicate its heavy pronunciation and M for indicating anusvara.

Examples
1. ಕಡನಲ -- kaaDinalli
2.ಅದಕಕ-- adakke
3.ಬಬಟ -- beeTe
4. ಕಕರಟ-- kyaaraT

Examining the above examples, we can understand the four different cases. And the software is designed to deal with all the cases. In the first and second example, we have a case of "Ottakshara" which are called consonant conjuncts in English. In the third example, we can see the use of "deergha". And in the fourth example, we can see the use of "Arkavottu" . ZWJ (zero width joiner) is nothing but a representation of arkavottu.

Consonants table

	U+0C9x	
5	ಕ	ka
6	ಖ	Kha
7	ಗ	ga
8	ಘ	Gha
9	ಙ	nga
a	ಚ	ca
b	ಛ	cha
c	ಜ	ja
d	ಝ	jha
e	ಞ	nya
f	ಟ	TTa

Vowels table

	U+0C8x	
2	◌ಂ	anusvara
3	◌ಃ	visarga
4	RESERVED	
5	ಅ	a
6	ಆ	aa
7	ಇ	i
8	ಈ	ii
9	ಉ	u
a	ಊ	uu
b	ಋ	r
c	ಌ	l
d	RESERVED	
e	ಎ	e
f	ಏ	ee

Consonant conjuncts table

Taking a letter and appending a consonant we get the third column. For example lets take the letter as “Ra”

	U+0CCx	output
0	◌ೀ	rii
1	◌ು	ru
2	◌ೂ	ruu
3	◌ೃ	rR
4	◌ೄ	rRR
5	Not Assigned	

6	◌ೆ	re
7	◌ೆು	ree
8	◌ೆೈ	Rai
9	Not Assigned	
A	◌ೆ◌ೂ	Ro
B	◌ೆ◌ೂು	Roo
C	◌◌ೌ	Rau
D	◌ೕ	r

Using these consonants, vowels and consonant conjuncts tables the software maps the corresponding Unicode of Kannada letter to English letter. Some Unicodes are reserved for constants. And we use the approach of removing the last letter and appending the next vowel.

Conditions to be followed

1) A single input consisting of a consonant that is followed by another consonant will be displayed without any modification to the string value of that input key.

ಕಗ will be translated as kaga.

2) An input consisting of a consonant and a vowel will be displayed by discarding the last character of the value string for the consonant key and appending the prefix to the vowel value.

ಸwill be split into ಸ and ◌ಿಿ .

This will be displayes by removing a from sa and appending I to it. The output will be si.

3) When a ottakshara is given as input, it consists of three to four unicode,where the first and the third will be consonants and the second and the fourth will be vowel. The final output will be obtained by removing the last character of the first and the third(if the fourth character is given) consonant and appending to the value of the key for the second and the fourth(if given) vowel.

ಪಪ has 3 characters :ಪ, ◌ಿ್ and ರ. Their equivalent pa and ra. Thus the ‘a’ in pa will be removed and the final word pra will be generated.

M is pronounced as the nasal sound corresponding to the row of the consonant that follows it in the given word. Thus,

Mk is pronounced as nGk,
Mc is pronounced as nYc,
MT is pronounced as NT,
Md is pronounced as nd,
Mb is pronounced as mb.

For the unclassified consonants, M is pronounced as m. when
preceded by the vowel a or aa, as hi when preceded by i, ii or
ai, as hu when preceded by u, uu or au, as as he when preceded by e or ee.

Conclusion

In this paper, we have presented our idea on development of different transliterature for Kannada language. The same method may not be applicable to all other language scripts because of the different grammatical structure of the words. Using the same method might cause some grammatical errors and may also change the meaning of the words. The main effort and challenge behind the development is to design the software by considering all test cases by forming derivative or compound words by putting together constituents each of which expresses a single definite meaning and rich features of language.

References

[1] Ekbal, A., Naskar, S. and Bandyopadhyay, S. 2006. A Modified Joint Source Channel Model for Transliteration. In Proceedings of the COLING-ACL 2006, 191-198, Australia.

[2] Ekbal, A. Naskar, S. and Bandyopadhyay, S. 2007. Named Entity Transliteration. Interna- tional Journal of Computer Processing of Oriental Languages (IJCPOL), V olume (20:4), 289-310, World Scientific Publishing Company, Singapore.

[3] Gurpreet Singh Josan & Jagroop Kaur (2011)' Punjabi to Hindi Statistical Machine Translaiteration', International Journal of Information Technology and Knowledge Management July-December 2011, Volume 4, No. 2, pp. 459-463.

[4] Amitava Das, Asif Ekbal, Tapabrata Mandal and Sivaji Bandyopadhyay (2009), 'English to Hindi Machine Transliteration System at NEWS', Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, page 80-83, Suntec, Singapore.

[5] Taraka Rama, Karthik Gali (2009), 'Modeling Machine Transliteration as a Phrase Based Statistical Machine Translation Problem', Language Technologies Research Centre, IIIT, Hyderabad, India.

[6] Vijaya MS, Ajith VP, Shivapratap G, and Soman KP (2008), 'Sequence labeling approach for English to Tamil Transliteration using Memory based learning', In Proceedings of Sixth International Conference on Natural Language processing.

[7] Antony P J, Ajith V P and Soman K P (2010), 'Feature Extraction Based English to Kannada Transliteration', Third International Conference on Semantic E-business and Enterprise Computing, SEEC-2010.

[8] Antony P J, Ajith V P and Soman K P (2010), 'Kernel Method for English to Kannada Transliteration', International Conference on-Recent Trends in Information, Telecommunication and Computing (ITC 2010), Paper is archived in the IEEE Xplore and IEEE CS Digital Library.

[9] ALA-LC Romanization Tables: Transliteration Schemes for Non-Roman Scripts. Randal K. Barry (ed.). Library of Congress, 1997. (http://lcweb.loc.gov/catdir/cpso/roman.html).

[10] Bright, William: "Kannada and Telugu Writing", in Peter T. Daniels & William Bright, eds. The World's Writing Systems. New York/Oxford, 1996.

[11] ISO 15919:2001. Information and documentation – Transliteration of Devanagari and related Indic scripts into Latin characters. International Standards Organization, 2001.

[12] Stone, Anthony P.: Transliteration of Indic Scripts: How to Use ISO 15919. 2002. (http://homepage.ntlworld.com/ stone-catend/trind.htm)