Transliteration among Indian Languages using WX Notation

Rohit Gupta
LTRC
IIIT, Hyderabad
India

Pulkit Goyal
Information Technology
IIIT, Allahabad
India

Sapan Diwakar
Information Technology
IIIT, Allahabad
India

rohit@research.iiit.ac.in {pulkit,sapan}@daad-alumni.de

Abstract

In this paper, we propose an algorithm to transliterate between several Indian languages. The main aim of the algorithm is to assist in the translation process by providing efficient transliteration. This algorithm works on Unicode transformation format of an Indian language. It then transliterates it into the Unicode transformation format of the target language. It does no sort of bilingual dictionary lookup of the word. It can be used to transliterate nouns (e.g. named entities) in the translation process as well as for transliterating some text into other language which is more suitable for the reader.

1 Introduction

With the advances in technology and availability of information in electronic format everywhere, it becomes important to provide this information to people as and when needed as well as in their native language. This calls for the development of a tool that can translate this information efficiently.

The translation process comprises of several steps, one of which is transliteration. By transliteration, we refer to the process of transcribing letters or words from one script to another script. In transliteration, word pronunciation is usually preserved. In some cases, it can also be modified according to its pronunciation in target language. Its main aim is to present the word in the destination languagefs script such that it is readable by the readers of the destination language (Surana and Singh, 2008).

The use of translation is even more necessary in a country like India that has immense diversity. There are different people who speak different languages in different regions of the country. Moreover, most of these languages have different scripts. Thus the application of translation is huge in India. Also, Indian languages are used by many people across the globe. Hindi, the most common of all the Indian languages is used by more than four hundred million people followed by Bengali (83m), Telugu (74m), Marathi (72m), Tamil (61m), Urdu (52m), Gujarati (46m), Kannada (38m) and Malayalam (33m) (Wikipedia, 2010).

We aim at providing an efficient algorithm for the transliteration process that is used to convert nouns (or other words) that are not present in the bilingual dictionary of the source language to the target language. Such software has other utilities as well when used as a standalone tool. One such utility of such software is in building an interface for users wherein they can type in an Indian Language using the more familiar QWERTY keyboard.

The idea is to allow users to type Roman letters and have them automatically transliterated into Indian Language. This is not as simple as it occurs because there is no direct mapping between Roman letters and Indian Language letters. There may be several combinations of characters which produce a single character in Indian Language or may produce vowel. The mapping that we have used in our work, is a more constrained and provides a rule set for writing a particular Indian Script in Roman letters which can then be converted into the Indian Script. This intermediate representation (known as WX notation explained in a greater detail later in the paper) also provides a way to convert the Indian Languages into one another considering that the phonetic pronunciation of the words in WX notation does not change with different scripts. This assumption is simplifying as well as holds true in most of the cases for Indian Languages. Our approach revolves around this concept of WX notation and inter-conversions between UTF notation of language to its WX notation and then from WX to UTF of the target language.

The paper is structured as follows. In Section 2, we briefly discuss the previous work carried out in this field. In Section 3, we describe our

methodology which is subdivided into three main modules as described in Sections 3.1, 3.2 and 3.3.

2 Previous Research

There have been several researches carried out in this area. Janarthanam, Sethuramalingam and Nallasamy (2008) proposed an algorithm that employs grapheme-based model. In their approach, the transliteration equivalents are identified by matching in a target language database based on edit-distance. The authors trained their tool with several names before the transliteration process. Surana and Singh (2008) present a different algorithm that eliminates the training phase. They used fuzzy string matching to account for the lack of training process. Karimi, Turpin and Scholer (2006) split the words into vowels and consonants to achieve transliteration. Their approach focuses on combining most probable combinations of vowels and consonants from source language to target language. A Statistical model for transliteration from English to Arabic words was implemented by Jaleel and Larkey (2003).

3 Methodology

Our algorithm works by converting the Unicode transformation format of source language to its corresponding WX notation taking into account the linguistic knowledge for each language. This WX notation is then converted to the Unicode transformation format of the target language to achieve transliteration. It utilizes the information stored in Unicode transformation format to automatically identify the source language. The target language, however, needs to be specified.

Before we begin with the description of the algorithm, let us first define what Unicode transformation format and WX notation are.

Definition 1: Unicode transformation format (UTF): It is the universal character code standard to represent characters. UTF-8 is an alternative coded representation form for all the characters in Unicode while maintaining compatibility with ASCII (Unicode Standard Version, 2003).

Definition 2: WX-Notation: WX notation is a transliteration scheme to denote a script in Roman script. It defines a standard for the representation of Indian Languages in Roman script. These standards aim at providing a unique representation of Indian Languages in Roman alphabet (Akshar et.al., 1995).

The WX notations for different Indian Languages are similar in their representation (See Table 1). We utilize this property for the development of our algorithm for transliteration.

Language	UTF-8	WX
Hindi	सचिन	Sacina
Bengali	সচিন	
Telugu	సచిన	
Punjabi	ਸਚਿਨ	
Malayalam	സച	
Kannada	ಸಚೀನ	

Table 1: Corresponding UTF and WX for various Indian Languages representing the word “Sachin”

Thus the problem of transliteration can now be divided into sub problems each of which can be addressed by designing converters for converting UTF to WX and WX to UTF for each language.

This method of conversion using an intermediate notation was necessary so as to limit the number of converters required for several languages (Using direct mapping, for 6 languages, we would have required 30 different transliteration tools whereas using the intermediate notation, we just need 6 tools for converting from UTF to WX and another 6 to convert back from WX UTF thus limiting the number of tools to just 12). Another benefit of this notation is that we can extend it to convert into other languages by simply adding 2 tools that could convert from UTF to WX and vice versa for that language.

3.1 Identifying Source Language

The first step in the transliteration process that we explain in the paper is to identify the source language. The source language of the given text can automatically be detected by analyzing the UTF characters. UTF characters follow a particular order in the representation of characters. All the characters of a particular script are grouped together. Thus we can identify which language/script is presented to the software by analyzing its character codes. UTF-8 characters are variable length. For Indian languages, these characters comprise of three bytes. Thus to detect the script of the UTF-8 characters, we analyzed the three bytes for different languages for some pattern. By comparing the code of second byte, the Indian Languages can be identified (See Table 2).

Language	Code for second byte
Hindi (hin)	164 or 165
Bengali (ben)	166 or 167
Telugu (tel)	176 or 177
Punjabi (pan)	168 or 169
Malayalam (mal)	180 or 181
Kannada (kan)	178 or 179

Table 2: Character Codes for UTF-8 representation of different Indian Languages

3.2 Converting UTF to WX

The next task is to convert the UTF form of language to the corresponding WX notation. This is achieved by using different converters for different languages. These converters are similar in their implementation with a few minor changes for each arising due to its linguistic rules. Firstly, we initialize the character maps which usually represent a many to one mapping from Roman characters to UTF. We then extract characters from the input string one by one. We then push the corresponding WX equivalents of these characters to the output string. We have to keep in mind about maintaining the efficiency of the algorithm so that searching for an element in the map is minimized. For this purpose, we have made a map that corresponds to the indices that we can obtain using UTF characters. Thus we donft need to search the map for UTF characters. Each UTF character has a different code and from that code, we can extract an index that points to its corresponding WX character. This finds the WX equivalent for each UTF character in constant time.

3.3 Converting WX to UTF

Once we obtain the WX notation for the given source text, the next step is to convert the WX notation to UTF of target language. This can be done using a similar mapping of Roman characters to UTF. Again we have to keep in mind about maintaining the efficiency of the algorithm so that searching for an element in the map is minimized. This is done by utilizing the ASCII codes of roman characters that are used to represent WX characters and then building the map as required. Thus WX to UTF conversion for each character is also achieved in constant time.

4 Results

In order to prove our algorithm, we compared the performance of our tool with the results provided on a test set by Linguists having knowledge of both the source as well as target language.

To evaluate our method, we tested our tool on a large corpus having 10k (approx. 240k words) sentences in Hindi. We then transliterated the complete corpus into each of the target languages one by one, results of which are listed in table 3.

Target Language	Accuracy
Hindi	95.8
Bengali	93.2
Telugu	90.0
Punjabi	92.9
Malayalam	85.2
Kannada	87.1

Table 3: Different Indian Languages and corresponding accuracy

The accuracy is based on the phonetic pronunciations of the words in target and source language and this was obtained from Linguistics having knowledge of both the languages.

यहाँ पहुंचना भी मुश्किल नहीं है | राणा सांगा को हराने के लिए मुगलों ने यहाँ कई बार आक्रमण किया जिनमें कई बार राणा सांगा घायल हुए | यहाँ का सबसे बआ आकर् समुद्र का सगम है | पर षंग्रहालय में षबषे ज्यादा देखने लायक बात राजा रवि वर्मा के चित्र हैं |

a) Input Text to transliteration module

यहाँ पहुंचना भी मुश्किल नहीं है | राणा सांगा को हराने के लिए मुगलों ने यहाँ कई बार आक्रमण किया जिनमें कई बार राणा सांगा घायल हुए | यहाँ का सबसे बआ आकर् समुद्र का सगम है | पर षंग्रहालय में षबषे ज्यादा देखने लायक बात राजा रवि वर्मा के चित्र हैं |

b) Output in Hindi

Figure 1: Results of transliteration module

Another important point to note in the transliteration module is its time efficiency. Since it may be used as a part of the complete translation tool, it has to perform its task very rapidly. Keeping this in view during our implementation, we now present the time taken by our tool.

For 100 words written in Devanagari (Hindi), the transliteration into Malayalam using our tool takes less than 0.100 seconds on an Intel Core 2 Duo, 1.8 GHz machine running Fedora.

5 Conclusion

In this paper, we present an algorithm for the efficient transliteration between Indian Languages. We presented a brief overview of UTF and WX notations and then our algorithm that involved transition from UTF to WX of source language and then back to UTF for target language.

6 Future Work

The algorithm presented in the paper is an efficient algorithm for transliteration and would be used in translation between Indian Languages. We are also exploring on how to make the mapping more efficient using automatic learning.

References

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing : A Paninian Perspective*. Prentice Hall of India.

Nasreen Abdul Jaleel and Leah S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of the twelfth international conference on Information and knowledge management*, New Orleans, LA, USA.

Srinivasan C. Janarthanam, Sethuramalingam S, and Udhyakumar Nallasamy. 2008. Named entity transliteration for cross-language information retrieval using compressed word format mapping algorithm. In *Proceedings of 2nd International ACM Workshop on Improving Non-English Web Searching*.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer. 2006. English to persian transliteration. In *SPIRE 2006*, pages 255–266.

Harshit Surana and Anil Kumar Singh. 2008. A more discerning and adaptable multilingual transliteration mechanism for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, India.

Addison Wesley The Unicode Consortium. 2003. The Unicode Standard, Version 4.0.

Wikipedia. 2010. Languages of india. http://en.wikipedia.org/wiki/languages_of_india. (accessed: April 21, 2010).