Challenges in Designing Input Method Editors for Indian Languages: The Role of Word-Origin and Context

Umair Z Ahmed **Kalika Bali** **Monojit Choudhury** **Sowmya VB**
Microsoft Research Labs India, Bangalore
{kalikab; monojitc}@microsoft.com

Abstract

Back-transliteration based Input Method Editors are very popular for Indian Languages. In this paper we evaluate two such Indic language systems to help understand the challenge of designing a back-transliteration based IME. Through a detailed error-analysis of Hindi, Bangla and Telugu data, we study the role of phonological features of Indian scripts that are reflected as variations and ambiguity in the transliteration. The impact of word-origin on back-transliteration is discussed in the context of code-switching. We also explore the role of word-level context to help overcome some of these challenges.

1 Introduction

Automatic Machine Transliteration finds practical use in various Natural Language Processing applications like Machine Translation, Mono lingual and Cross lingual information retrieval. Backward transliteration – the reverse process of converting a transliterated word into its native script, has been employed as a popular mechanism for multilingual text-input (Sandeva et al, 2008; Ehara and Tanaka-Ishii, 2008). This has given rise to many Input Method Editors (IME)s that allow the use of a normal QWERTY keyboard to input text in non-Roman scripts like Japanese, Chinese, Arabic and several Indic languages

Roman transliteration is widely used for inputting Indian languages in a number of domains. A lack of standard keyboards, a large number of scripts, as well as familiarity with English and QWERTY keyboards has given rise to a number of transliteration schemes that are used for generating Indian language text in roman transliteration. Some of these are an attempt to standardise the mapping between the Indian language script and the Roman alphabet, e.g., ITRANS (Chopde, 1991) but mostly the users define their own mappings that the readers can understand given their knowledge of the language. A number of Indian language IMEs exist that employ either standardised mappings or try to account for user variations through rules, statistical methods or a combination of both. These Machine Transliteration systems may be used as Input Method Editors (IMEs) for desktop application, e.g., Baraha[1] or as web applications, e.g., Google Transliterate [2] and Quillpad[3]. Microsoft Indic Language Input Tool (MSILIT)[4] supports both a desktop as well as a web-based version. While all the above systems are popular and seem to serve their purpose adequately, there has not been any systematic evaluation to identify and address common problems that they may face, either specific to the languages concerned or due to the process of back-transliteration. As Knight and Graehl (1998) point out back-transliteration is "less forgiving" than forward transliteration for there may be many ways to transliterate a word in another script but there is only one way in which a transliterated word can be rendered back in its native form. For example, "London"

[1] http://www.baraha.com/

[2] http://www.google.com/transliterate/

[3] http://quillpad.in/hindi/

[4] http://specials.msn.co.in/ilit/Hindi.aspx

Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011), pages 1–9,
Chiang Mai, Thailand, November 13, 2011.

may be transliterated as "लंदन" or "लण्डन" in Hindi but any back-transliteration can generate only one correct case, that is, "London".

One reason for the absence of any meaningful evaluation is the lack of a standard dataset. The NEWS workshop (Li et al, 2009) made available training and test data in three Indian Languages – Hindi, Tamil and Kannada, but as this was constrained to named entities, it is of limited use for evaluating a general purpose transliteration system. Sowmya et al (2010) describes the creation of a dataset for three Indian languages, viz., Hindi, Bangla and Telugu transliterated into Roman alphabet. The availability of this dataset has made it possible to evaluate transliteration based Input mechanisms on common grounds and identify areas for improvement.

In this paper, we use the dataset described in (Sowmya et al, 2010) to evaluate two back-transliteration based Indian language IMEs to identify some of the common challenges faced by such systems. We discuss in some details the errors caused due to a) phonological variation or the variability in transliteration caused by the phonological properties of the source language, and b) word-origin – the transliteration of words or origin other than the source language. We also discuss how word-level context can help resolve some of these issues. While the examples presented here are mainly from Hindi, many of the experiments were also repeated for Telugu and Bangla, and can be generalized across these languages.

The rest of the paper is organized as follows: The next section presents evaluation data, methodology and a top-level error-analysis. In Section 3, we discuss the various phonological variations that cause ambiguous transliterations. In Sec 4, the role of word origin on back-transliteration is discussed. Section 5 discusses the impact of word-level context on back-transliteration. Further issues and possible future directions are discussed in Section 6.

2 An Evaluation of Indic IMEs

Two of the publicly available systems were chosen for evaluation on the same test data. Both the systems, as is usual for most Indic language IMEs, take continuous Roman input and convert it automatically into the relevant Indic language string after pause or punctuation. The user can select from a list of other possible options by a right-click on the relevant word. The aim of this evaluation was neither competitive nor to discover which system was better but to uncover common issues that plague back-transliteration based IMEs. The assumption was that an in-depth analysis of the common errors produced would help in a better understanding and ultimately in better systems. Further, the systems remain a black-box for this study as we do not have access to the internal models and algorithms being used and are therefore labeled as System A and B to mask their identity.

2.1 Data

The evaluation data for the three languages, Bangla, Hindi and Telugu, was collected through a series of user experiments. The methodology for the design and creation of the dataset is described in greater detail in Sowmya et al (2010). However, it is important to point out that these user experiments were conducted in three modes:

1. *Dictation*: Users were asked to listen to some speech files in their respective languages and transcribe them using Roman script. This was done on around 20 users per language, with 75 sentences per user, of which 50 were common to all the users. This common set was used to capture the variations in spellings across users.
2. *Topic writing*: Users were given a list of topics to choose from and write a few lines on two of them in their language, but using Roman script. Around 100 words were collected per user in this mode.
3. *Chat*: Users were asked to chat with another person for a few minutes using an Instant Messenger. These were general informal chat sessions, where the users used Roman script to chat about any topic of their choice in their own language. Around 50 words were collected per user.

Around 20000 words were collected per language, and the gold standard transcriptions were obtained manually.

2.2 Evaluation Methodology and Results

The dataset described above was used for evaluating the two commercial IME systems. Roman transliterations for all the words for each language were input to obtain the Top-1 result from both the engine. The output from the systems was analyzed quantitatively as well as manually to identify common patterns of errors. The accuracies of both the systems were found to be comparable across the number of unique words in the test set (Type), as well as the total number of words counting multiple occurrences of the same type (Token). Type level accuracies for the systems were around 55%, whereas the Token level accuracies lay between 75-78%.

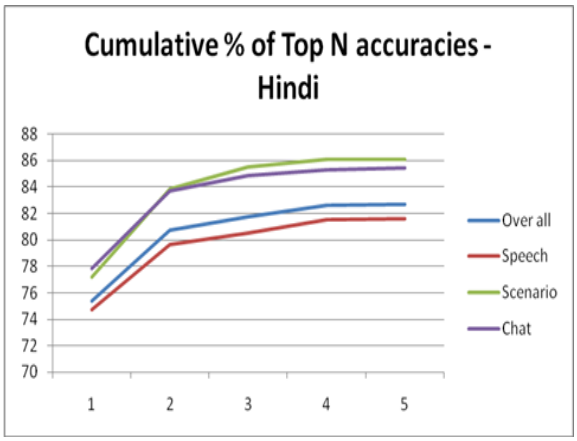

Figure 1: Cumulative Top-N token accuracy percentages for Hindi

Figure1 shows the cumulative Top-N token accuracy percentages for Hindi. It shows the percentage of words which the systems got right within Top-N (N varying from 1 to 5). As mentioned before, the test set had three kinds of data: words collected through speech transcription (speech), by chatting with the users (chat) and through the users writing a few lines on different topics (Topic writing). It may be noted that the Scenario data performed better over Chat data and Speech data. The performance was relatively poorer with Speech and this might be attributed to the noise in speech which made the users enter the wrong words. Bangla and Telugu showed similar trends.

2.3 Error Analysis

We have performed a manual error analysis on a random sample of around 400 words under each category. The errors observed may be classified as below:

- *Abbreviations*: When a given acronym or abbreviation in English is transliterated as a native word in Hindi, instead of being spelt out. For example, *CBI* is transliterated as कबी [kəbi] (Top-1) not as सीबीआइ [si#bi#ɑ:ɪ]
- *Code-mixing*: The interspersing of Hindi text with other language words, usually English, is known as code-mixing. This results in an English word being transliterated as a native Hindi word. For example, the word "missile" is transliterated as मिस्सीले [mɪs:ile] (Top-1) not मिसाइल [mɪsɑ:ɪl].
- *Misspellings:* The word is spelt incorrectly due to a typing error or other reasons. For examples, spelling *tayyariyon* as *tyyariyon* gives as त्य्यरियों [təj:ərɪjõ] as output instead of तैयारियों [təɪjɑ:rɪjõ].
- *Phonological variations:* All words that do not fall into the above three classes were studied for the variation in the way certain phonological features are represented in the two scripts. The mapping of certain features like aspiration or vowel length that is marked on the script for an Indian language like Hindi on the Roman alphabet can result in ambiguous transliteration. For example, the voiced aspirated velar in घर [ghər] may be represented as "*ghar*" or "*gar*". Similarly, the difference between the long and the short vowels is also not necessarily maintained in the Roman transliterations. Hence, मन [mən] and मान [mɑ:n] can both be transliterated as "*man*".
- *Others:* There are other sources of errors like incorrect transliteration in the gold standard, named entities being transliterated as normal text, and so on. These are either negligible in number or are not an error generated by the system

Table 1 shows the distribution of errors over these various categories for the three languages Hindi, Bangla and Telugu. While the absolute numbers vary across the languages, they clearly show that tackling these issues would go far towards increasing the accuracies of these systems.

Class	**% of Occurrence**		
	H	B	T
Abbreviations	6.5 %	-	10.31%
Code Mixing	9.25%	2 %	17.71%
Misspellings	13.25%	8%	25.71%
Phonological variation	13.75%	36%	14.28%

Table 1: The classification of errors for Hindi(H), Bangla (B) and Telugu (T)

3 Phonological Variations

The representation of sounds of one language using the script of another can lead to a many-to-many mapping between the sounds and the letters of the two scripts. This in turn results in many ambiguities in transliterations due to a many-to-many mapping between the orthographic units of the two scripts. For instance, the letter '*t*' is used to transliterate the Hindi sounds /ʈ/ (retroflex) or /t/ (dental plosive), which are represented in the Hindi script (Devanagari) as two distinct characters ट and त, respectively. Thus, if the input Roman string contains a '*t*', then depending on the context it can be transliterated as either ट or त, as there is no clear orthographic distinction in Roman script for the corresponding phonemic distinction in Devanagari.

On the other hand, the Devanagari character त is usually transliterated as '*t*' or '*th*', while '*th*' is also used commonly for representing the aspirated counterpart of त, that is, थ. These individual differences are not arbitrary and users are usually cognizant of the linguistic explanations behind them.

Thus, the problem of many-to-many mapping between characters during in the context of Indian language IMEs requires detailed discussion as many of these ambiguities are systematic in nature, have valid linguistic reasons and hold good for all Indian languages. In this section we categorize the different kinds of systematic phonological ambiguities which arise in the context of Indic language IMEs due to certain unique features of Indic scripts and their divergence from the Roman scripts.

3.1 Retroflex and Aspiration

Many Indian languages like Hindi, Bangla and Telugu show a two way contrast between voicing and aspiration to obtain a four member stop consonant series. Aspiration in consonants is generally assumed to be represented in Roman alphabet as the addition of "*h*". Thus, the aspirated velar stop, ख is represented by "*kh*", the aspirated voiced velar घ as "*gh*".

There are hence, four retroflex stop consonants represented by the characters "ट, ठ, ड, ढ" in Devenagari. As mentioned in the above section, our analysis clearly indicates that "*t*" is used to represent the unaspirated retroflex consonant ट (98.51%) and its dental counterpart त (96.55%). While "*th*" is almost always used for the aspirated versions: ठ (89.89%) and थ (95.56%). This trend is observed across the voiced counterparts as well.

In Telugu, however, the results are less straightforward. While "*th*" is used for the retroflex unaspirated ట (99.9%), its use to represent the other three consonants retroflex aspirated ఠ(70%), dental unaspirated త (68.85%) and the dental aspirated ధ (62.29%) shows a lot more variation. Table 2 shows the variation of the retroflex and the dental voiceless stops with "*t*" and "*th*" across the two languages.

This trend holds true to a large extent for all aspirated consonants. Thus, though to a large extent "*h*" is used to indicate aspiration for Hindi, Telugu presents a different result with the around 38% of the aspirated consonants spelt without "h" and 15% of the cases where an unaspirated consonant was spelt with "*h*".

Hindi				
	ट	ठ	त	थ
Spelt "*t*"	98.51%	10.10%	96.44%	4.43%
Spelt "*th*"	1.49 %	89.89%	3.55 %	95.56%
Telugu				
	ట	ఠ	త	థ
Spelt "*t*"	0.06%	70%	68.8%	37.7%
Spelt "*th*"	99.93%	30%	31.15%	62.29%

Table 2: Spelling variations for retroflex voiceless stop series in Hindi and Telugu

3.2 Vowels, Diphthongs and Semi-vowels

As length cannot be represented in the Roman alphabet as a single character, the transliteration of short and long vowels is another major source of considerable ambiguity for Indian languages. Thus, for any long-short vowel pairs which are phonemic in nature, and for which two different characters occur in Indian languages, the options available to the users are to either use the same single vowel or use two o characters to indicate length. For example, for the Hindi words सुना [sʊnɑ] "heard", and सूना [sunɑ] "lonely", a user may use two different transliterations: "*suna*" and "*soona*", respectively, or they might use the same transliteration, "*suna*" to represent both the words. Our analysis of the data shows that over 73% of the time, long vowels are spelt the same way as their shorter counterparts in Hindi. The other two languages show similar trends.

As far as diphthongs are concerned, all languages do use a sequence of the component vowels to represent diphthongs, however, there is much variability in this. For example, in Hindi, the diphthong "औ" [əʊ] may be transliterated as "*au, ou, o,aw, ow*". Similar multiple mappings are used for semi-vowels as well. Further, there is some overlap between the representation of diphthongs and semi-vowels as well. For example, the semi-vowel య in Telugu is represented by "*y, ya, yi, ey, ay*", and the diphthong ఐ by "*i,ai,ei,a,ey,y,ay*"

3.3 Fricatives and Affircates

The fricatives in Indian languages pose two main problems for transliteration:

a) The presence of similar graphemes used for stop consonants. E.g., The characters ज़ [z] and फ़ [f] in Hindi are most often confused even in the native script with their corresponding stop consonants, ज [ʤ] and फ [ph], and this often leads them to be represented by the same Roman alphabets, "*j*" and "*f*" respectively,

b) The sibilants often have an overlap within the series, say, between Hindi retroflex ष [ʂ] and palatal श [ʃ], both represented by "sh". Further, the same Roman transliteration may be used for palatal affricates and sibilant fricatives, for example, the Hindi alveolar fricative स [s] is a potential source of ambiguity since it is spelt both as "*s*" as well as "*c*", the latter being used to spell the palatal affricate consonant च [ʧ].

4 Errors due to Foreign Origin Words

The results in Table 1 indicate that a prominent source of error in IMEs for Indic languages is the inability of the transliteration systems to handle abbreviations and code-mixing, both of which are foreign origin words. The errors result from an implicit assumption made by the system that the user is typing an Indic (say Hindi) origin word in Roman. Therefore, if the input is "WHO", the system tries to transliterate it as व्हो [wʱo], which would be the case if we do a letter by letter mapping assuming the standard rules for English-Hindi back-transliteration, instead of more appropriate results हु [hʊ] (transliteration of the English word "who") or डब्ल्यूएचओ [dəbʌju#etʃ#o] (transliteration of the Abbreviation WHO).

These errors is not restricted to abbreviations, acronyms or English origin words, but can arise whenever the input word is of foreign origin, and the input form is not a transliterated Roman form, such as non-native names (e.g., "Michael"). However, we observe that abbreviations and code-mixing of English words (including English names) are the most common types of foreign origin words that lead to system er-

rors. Figure 2 shows a taxonomy of foreign and mixed origin words that is relevant to the present discussion.

4.1 Abbreviations

In this class we include both Acronyms (e.g., LASER), that is, words created from combining sub-parts of two or more words, as well as Initialism (e.g., BBC), where a word is created by taking the first letters or initials of a phrase. Both could be considered a special case of blends or words formed from partial content of existing words (Cook and Stevenson, 2010).

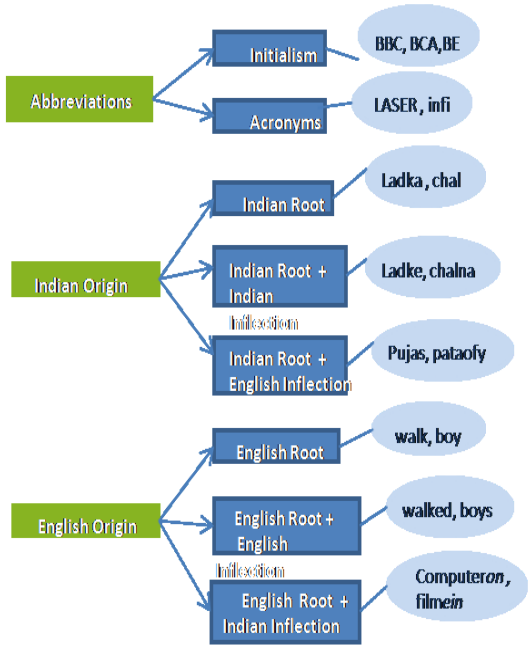

Figure2: Different classes and sub-classes of the input text

Most users writing in transliterated Hindi preserve these forms in text. Thus, instead of transliterating the above example of initialism as "bee bee see" in Hindi, they use the original "BBC". The system, expecting a word of Hindi-origin, might exclude the right result from the top relevant results in such cases.

Most of the abbreviations are incorrectly transliterated by the IME systems, except for some very common ones. It seems that instead of generic technique for identifying abbreviations, the Indic IMEs presently list-lookup based approach where very common abbreviations are explicitly listed out. While it might be a hard task to automatically identify abbreviations with high accuracy, IMEs can at the least, provide the transliteration of the abbreviation as one of the options. Note that abbreviations can be very easily transliterated using a set of mapping rules.

4.2 Code-mixing

Code-mixing refers to instances where lexical items and grammatical features of two languages appear in a single utterance (Muysken 2000) and is a common and well-studied phenomenon in a bilingual society where it exists at all except the most extremely formal form of spoken and written language (Romaine and Kachru, 1992). In the context of transliterated text, code-mixing can be viewed as a form of noise caused due to multilinguality in text. Sowmya et al (2010) study the extent of code mixing and its patterns in Hindi, Bangla and Telugu transliterated text. Their analysis shows that though the context (formal versus informal) does play a vital role in this, and the strategies may vary from language to language, in general, all three languages show similar trends in code-mixing.

Error analysis shows that code-mixing can occur at different levels of language structure, that is, the mixing can be at the level of words as well as inflection. Thus, we can have code-mixing where English words are interspersed in Hindi text. For example, "*mere **friends** kal **train** se ayenge*" (my friends will come tomorrow on the train) where "friends" and "train" appear in their English form. If we assume that *friends* and *train* are of Hindi origin words, then following the commonly observed rules of transliteration we would get फ्रिएंड्स [frɪeɳḍs] or फ्रिएन्ड्स [frɪeɳɳḍs] and ट्रैन [ʈrɛn] respectively, while the correct transliterations should have been either फ्रेंड्स [freɳḍs] and ट्रेन [treɳ], or their original Roman spellings (which might look a little informal style of writing, but not unsual).

It should be possible to handle foreign origin words through a two step process: first, identification of the origin of the word using a classifier, and second, transliteration of non-native words using a different statistical transliterator or by using different sets of rules. In fact, Khapra and Bhattacharyya (2009) and

Chinnakotla et al (2010), have both reported improved accuracies in transliteration tasks through origin detection.

There are also instances of an English word with Hindi inflections, as in "computer*on*" [kəmpjuʈərõ] (computer + Hindi plural marker *on*), and Hindi words with English inflections – "*sadak*s" [səʈəkõ] (*sadak* meaning road in Hindi + English plural marker *s*). Such cases are rare in the dataset, and it might be much harder a challenge to automatically identify morpheme level code-mixing and subsequent transliteration. We do not know of any previous studies addressing this issue, and therefore, it would be a very interesting topic of future research.

5 Effect of Word-level Context

Word-level context can provide useful information to resolve some of the errors due to phonological and spelling variation as well as misspellings reported in Section 2. Consider the following example: The Roman string *choti* can be transliterated as छोटी [tʃʰoʈi] (small) or चोटी [tʃoʈi] (peak or braid). Thus, it is a case of genuine ambiguity resulting from phonological variations. छोटी has a higher unigram frequency than चोटी, and therefore, most IMEs output the former as the first option for the input 'choti'. However, in the context of "*himalaya ki choti"* the most likely back-transliteration would be "हिमालय की चोटी" [hɪmələjə#ki#tʃoʈi] meaning "peak of the Himalayas", and not "हिमालय की छोटी" [hɪmələjə#ki#ʧʰoʈi] meaning "Himalaya's small". Similarly, in the sentence *mujhe aaj kam ko jaana hai* [moʊʤʰe#ɑ:ʤ#kɑm#ko#ʤɑnɑ#hɛ] ("I have to go to work today"), the system transliterates *kam* as "कम" [kəm] (with a short vowel) meaning "less" instead of "काम" [kɑm] (with a long vowel) meaning "work".

On the contrary, none of the commercial IMEs studied during this work incorporates context aware transliteration. The top ranked output for *choti* is छोटी irrespective of the context. In other words, the back-transliteration based IMEs transliterate each word independently. Clearly, word-level context is crucial in back-transliteration. IMEs incorporating context-aware transliteration schemes can help the user type faster by getting the correct suggestion on top more often saving a few clicks, and thereby improving the user experience.

We conducted language model (LM) based experiments in order to obtain some estimate of the possible benefits of a simple word-level language model in IMEs. The aim of conducting the LM experiments is not to discover which kind of models and algorithms are best suited for IME; rather our aim is to establish the fact that language models, even in their simplest avatars, can indeed help improving performance of Indian language IMEs. We present some first cut experimental results in this direction.

Standard noisy channel model, *p(target|source)=p(source|target)p(target)*, where the first part is the channel model and the second part is the language model, cannot be used for our experiments. While we can compute *p(target)* using standard language model features, we do not have the probabilities of the channel model, because we are using a black-box transliteration engine which returns a ranked list of candidates for a given source word.

In our experiments, we suppose for the input string "*w1 w2 w3*", the ranked outputs for *w3* by an IME are *h1*, *h2*, *h3*, *h4* and *h5*. We assume that we know the correct outputs for *w1* and *w2*, say *w1** and *w2**, which can be obtained from the gold standard transliterations. Then we search for the strings "*w1* w2* hi*" (where *i* is between 1 and 5) on the Web using a commercial search engine and re-rank the outputs based on the number of webpages returned. We started by exploring n-gram models learnt from a Hindi corpus consisting of newspaper articles. However, on analyzing the results we discovered that the n-gram statistics for the dataset used in Sowmya et al (2010), especially the chats and blog, follow a very different distribution from those we observe in the newspaper corpus. Therefore, we decided to obtain the n-gram statistics from the Web.

This web count based re-ranking experiment on the Sowmya et al (2010) data shows a relative improvement of 10-20% for the top-1 accuracy. The values are summarized in table 3.

The absolute improvement figures are small because as evident from Fig. 1, in most of the cases, whenever the current transliteration is generated by the system, it is usually at top rank.

Type of Data	Accuracy without LM	Absolute Accuracy with LM	Relative improvement in Accuracy
Chat	78.86%	80.22%	21.95%
Scenario	78.56%	79.52%	13.10%

Table 3: Accuracy improvement using web-search based LM

Therefore, we report the relative improvement in accuracy which shows the percentage of cases where the correct transliteration was present in top 5, but not in top 1, and the re-ranking based approach has been able to pull it to top-1 rank. Thus, there are further opportunities for improving user experience in Indic language IMEs through context aware back-transliteration.

6 Discussion

In this paper, we have attempted to do a systematic evaluation of the common challenges faced for designing back-transliteration based IME systems for Indic-languages through a thorough analysis of the errors produced. We focused on a few of the main sources of errors that occurred due to the inability of the systems to deal with a) phonological variations, and b) words of different origin

As our error-analysis showed, spelling variation can occur because of certain phonologically motivated phenomenon. This is primarily a result of trying to map a 50+ set of phonemes of the Indian languages on to a 26 character set of the Roman alphabet. This results in many-to-many mappings between the two scripts and conventions which may be region or language specific. Thus, a Hindi speaker might transliterate a dental stop as "t" while a South Indian might use the same character to denote a retroflex. Some other conventions might be specific to an individual. Hence, the ability to identify user-specific patterns or mapping could help tackle errors induced by variation and user-adaptation could go a long way in achieving a more accurate back-transliteration based IME with a much higher utility.

Code-mixing and abbreviations add another dimension of transliteration errors, and one that is largely ignored by the current IMEs. In the Indian context, a module to handle at the very least, English words, would go a long way in resolving this problem. Given the extent of code-mixing in Indian languages this is a relevant research problem.

Lastly, we have shown that a context-aware Indic language IME that takes into account a Language Model at word-level can possibly help address some of these challenges. A re-ranking based approach can be further explored to not only boost accuracies but design innovative ways to improve user experience.

References

Animesh, N., Ravikiran Rao, B., Pawandeep, S Sudeep, S. And Ratna, S. (2008). Named Entity Recognition for Indian Languages. *In proceedings of IJCNLP 2008 workshop on NER for South and South-East Asian languages.*

Chinnakotla, M. K., Damani, O. P., and Satoskar, A. 2010. Transliteration for resource-scarce languages. *ACM Trans. Asian Lang. Inform. Process. 9, 4, Article 14 (December 2010)*

Chopde, A. (1991-2001), Printing Transliterated Indian Language Documents- ITRANS V 5.30. http://www.aczoom.com/itrans/html/idoc/idoc.html

Cook, P. And S. Stevenson, (2010). Automatically identifying the source words of lexical blend in English. *Computational Linguistics 36(1):129-149*

Ehara, Yo. and Kumiko Tanaka-Ishii. (2008). Multilingual text entry using automatic language detection. *In proceedings of IJCNLP 2008*.

Khapra, Mitesh M., and Pushpak Bhattacharyya. 2009. Improving Transliteration Accuracy Using Word-Origin Detection and Lexicon Lookup. *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009),* ACL.

Li, H., Kumaran, A., Zhang, M., and Pervouchine, V. (2009). Report on NEWS 2009 Machine Transliteration shared task. *In proceedings of ACL-IJCNLP 2009Named EntitiesWorkShop.*

Knight, K. and Graehl J. (1998). Machine Transliteration. *Computational Linguistics (Vol: 24. Issue 4.*

Muysken, P. (2000). *Bilingual Speech-A typology of code-mixing*. Cambridge University Press, Cambridge.

Sandeva, G., Hayashi, Y., Itoh,Y., and Kishino, F. (2008). SriShell Primo: A Predictive Sinhala Text Input System. *In proceedings of IJCNLP 2008 workshop on NLP for less privileged languages*

Scott McCarley, J. (2009), Cross language name matching. *In proceedings of ACM-SIGIR2009*

Sowmya V.B., Monojit Choudhury, Kalika Bali, Tirthankar Dashupta and Anupam Basu. (2010). Resource creation for training and testing of transliteration systems for Indic languages. *To be presented at Language Resources and Evaluation Conference (LREC), 2010.*

Romaine, Suzanne and Braj Kachru. 1992. "Code-mixing and code-switching." In T. McArthur (ed.) *The Oxford Companion to the English Language*. Oxford University Press. pp. 228-229