A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages

Harshit Surana
Language Tech. Research Centre
IIIT, Hyderabad, India
surana.h@gmail.com

Anil Kumar Singh
Language Tech. Research Centre
IIIT, Hyderabad, India
anil@research.iiit.ac.in

Abstract

Transliteration is the process of transcribing words from a source script to a target script. These words can be content words or proper nouns. They may be of local or foreign origin. In this paper we present a more discerning method which applies different techniques based on the word origin. The techniques used also take into account the properties of the scripts. Our approach does not require training data on the target side, while it uses more sophisticated techniques on the source side. Fuzzy string matching is used to compensate for lack of training on the target side. We have evaluated on two Indian languages and have achieved substantially better results (increase of up to 0.44 in MRR) than the baseline and comparable to the state of the art. Our experiments clearly show that word origin is an important factor in achieving higher accuracy in transliteration.

1 Introduction

Transliteration is a crucial factor in Cross Lingual Information Retrieval (CLIR). It is also important for Machine Translation (MT), especially when the languages do not use the same scripts. It is the process of transforming a word written in a source language into a word in a target language without the aid of a resource like a bilingual dictionary. Word pronunciation is usually preserved or is modified according to the way the word should be pronounced in the target language. In simple terms, it means finding out how a source word should be written in the script of the target languages such that it is acceptable to the readers of the target language.

One of the main reasons of the importance of transliteration from the point of view of Natural Language Processing (NLP) is that Out Of Vocabulary (OOV) words are quite common since every lexical resource is very limited in practical terms. Such words include named entities, technical terms, rarely used or 'difficult' words and other borrowed words, etc. The OOV words present a challenge to NLP applications like CLIR and MT. In fact, for very close languages which use different scripts (like Hindi and Urdu), the problem of MT is almost an extension of transliteration.

A substantial percentage of these OOV words are named entities (AbdulJaleel and Larkey, 2003; Davis and Ogden, 1998). It has also been shown that cross language retrieval performance (average precision) reduced by more than 50% when named entities in the queries were not transliterated (Larkey et al., 2003).

Another emerging application of transliteration (especially in the Indian context) is for building input methods which use QWERTY keyboard for people who are more comfortable typing in English. The idea is that the user types Roman letters but the input method transforms them into letters of Indian language (IL) scripts. This is not as simple as it seems because there is no clear mapping between Roman letters and IL letters. Moreover, the output word should be a valid word. Several commercial efforts have been started in this direction due to the lack of a good (and familiar) input mech-

anism for ILs. These efforts include the Google Transliteration mechanism[1] and Quilpad[2]. (Rathod and Joshi, 2002) have also developed more intuitive input mechanisms for phonetic scripts like Devanagari.

Our efforts take into account the type of the word, the similarities among ILs and the characteristics of the Latin and IL scripts. We use a sophisticated technique and machine learning on the source language (English) side, while a simple and light technique on the target (IL) side. The advantage of our approach is that it requires no resources except unannotated corpus (or pages crawled from the Web) on the IL side (which is where the resources are scarce). The method easily generalizes to ILs which use Brahmi origin scripts. Our method has been designed such that it can be used for more conventional applications (MT, CLIR) as well as for applications like building an input mechanism.

Much of the work for transliteration in ILs has been done from one Indian script to another. One of the major work is of Punjabi machine transliteration (Malik, 2006). This work tries to address the problem of transliteration for Punjabi language from Shahmukhi (Arabic script) to Gurmukhi using a set of transliteration rules (character mappings and dependency rules). *Om* transliteration scheme (Ganapathiraju et al., 2005) also provides a script representation which is common for all Indian languages. The display and input are in human readable Roman script. Transliteration is partly phonetic. (Sinha, 2001) had used Hindi Transliteration used to handle unknowns in MT.

naukri (A popular domain name)	722,000
nokri (domain name)	19,800
naukari	10,500
naukary (domain name)	5,490
nokari	665
naukarii	133
naukaree	102

Table 1: Variations of a Hindi Word nOkarI (job). The numbers are pages returned when searching on Google.

[1] www.google.co.in/press/pressrel/news_transliteration.html
[2] www.quillpad.com

Aswani et. al (Aswani and Gaizauskas, 2005) have used a transliteration similarity mechanism to align English-Hindi parallel texts. They used character based direct correspondences between Hindi and English to produce possible transliterations. Then they apply edit distance based similarity to select the most probable transliteration in the English text. However, such method can only be appropriate for aligning parallel texts as the number of possible candidates is quite small.

The paper is structured as follows. In Section-2, we discuss the problem of a high degree of variation in Indian words, especially when written in Latin script. In Section-3, we explain the idea of using information about the word origin for improving transliteration. Then in Section-4 we describe the method that we use for guessing the word origin. Once the word origin is guessed, we can apply one of the two methods for transliteration depending on the word origin. These two methods are described in Section-5 and Section-6, respectively. Fuzzy string matching, which plays an important role in our approach, is described in Section-7. In Section-8 we put together all the elements covered in the preceding sections and explain the Discerning Adaptable Transliteration Mechanism. Section-9 presents the evaluation of our approach in comparison with two baseline methods, one of which uses knowledge about word origin. Finally, in Section-10 we present the conclusions.

2 Variation in Indian Words in Latin Script

Since the purpose of our work is not only to transliterate named entities but to be useful for applications like input mechanisms, we had to consider some other issues too which may not be considered directly related to transliteration. One of these is that there is a lot of spelling variation in ILs. This variation is much more when the IL words are written using the Latin script (Table-1). In other words, the amount of ambiguity is very high when we try to build a system that can be used for purposes like designing input mechanisms, instead of just for transliteration of NEs etc. for MT or CLIR. One reason for very high variation in the latter case is that unlike Romaji for Japanese (which is taught in

schools in Japan), there is no *widely adopted* transliteration scheme using the Latin script, although there are a number of standard schemes, which are not used by common users. At present the situation is that most Indians use Indian scripts while writing in ILs, but use the Latin script when communicating online. ILs are rarely used for official communication, except in government offices in some states.

3 Word Origin and Two Ways of Transliteration

Previous work for other languages has shown that word origin plays a part in how the word should be transliterated(Oh and Choi, 2002; May et al., 2004). Llitjos and Black (Llitjos and Black, 2001) had shown that the knowledge of language origin can substantially improve pronunciation generation accuracy. This information has been used to get better results (Oh and Choi, 2002). They first checked whether the word origin is Greek or not before selecting one of the two methods for transliteration. This approach improved the results substantially. However, they had used a set of prefixes and suffixes to identify the word origin. Such an approach is not scalable. In fact, in a large number of cases, word origin cannot be identified by using list of affixes.

For ILs, we also define two categories of words: words which can be roughly considered Indian and those which can be roughly considered foreign. Note that 'Indian' and 'foreign' are just loose labels here. Indian words, which include proper nouns and also common vocabulary words, are more relevant in applications like input methods. Two different methods are used for transliterating, as explained later.

4 Disambiguating Word Origin

Previously (Llitjos and Black, 2001) used probabilities of all trigrams to belong to a particular language as an measure to disambiguate word origins. We use a more sophisticated method that has been successfully used for language and encoding identification (Singh, 2006a).

We first prepare letter based 5-gram models from the lists of two kinds of words (Indian and foreign). Then we combine n-grams of all orders and rank them according to their probability in descending order. Only the top N n-grams are retained and the rest are pruned. Now we have two probability distributions which can be compared by a measure of distributional similarity. The measure used is symmetric cross entropy or SCE (Singh, 2006a).

Since the accuracy of identification is low if test data is very low, which is true in our case because we are trying to identify the class of a single word, we had to extend the method used by Singh. One major extension was that we add word beginning and ending markers to all the words in training as well as test data. This is because n-grams at beginning, middle and end of words should be treated differently if we want to identify the 'language' (or class) of the word.

For every given word, we get a probability about its origin based on SCE. Based on this probability measure, transliteration is performed using different techniques for different classes (Indian or foreign). In case of ambiguity, transliteration is performed using both methods and the probabilities are used to get the final ranking of all possible transliterations.

5 Transliteration of Foreign Words

These words include named entities (George Bush) and more common nouns (station, computer) which are regularly used in ILs. To generate transliteration candidates for such words, we first try to guess the word pronunciation or use a lookup dictionary (if available) to find it. Then we use some simple manually created mappings, which can be used for all Indian languages. Note that these mappings are very few in number (Figure-1 and Figure-2) and can be easily created by non-linguistically trained people. They play only a small role in the method because other steps (like fuzzy string matching) do most of the work.

For our experiments, we used the CMU speech dictionary as the lookup, and also to train pronunciation estimation. If a word is not in the CMU dictionary, we estimate the word pronunciation, as explained later.

We directly map from English phonemes to IL letters. This is based on our observation that a foreign word is usually transliterated in almost the same way as it is pronounced. Almost all English phonemes can be roughly mapped to specific letters (representing phonemes, as IL scripts are phonetic in na-

ture) in ILs. Similar observations have been made about Hindi by Su-Youn Yoon, Kyoung-Young Kim and Richard Sproat (Yoon et al., 2007). We have prepared our own mappings with help from native speakers of the languages concerned, which is relatively quite a simple task since the letters in Indic scripts correspond closely with phonemes.

6 Transliteration of Indian Words

These words include (mainly Indian) named entities of (e.g. Taj Mahal, Manmohan Singh) and common vocabulary words (common nouns, verbs) which need to be transliterated. They also include words which are spelled similar to the way Indian words are spelled when written in Latin (e.g. Baghdad, Husain). As stated earlier, this class of words are much more relevant for an input method using a QWERTY keyboard.

Since words of Indian origin usually have phonetic spellings when they are written in English (Latin), the issue of pronunciation estimation or lookup is not important. However, there can be many possible vowel and consonant segments which can be formed out of a single word. For example 'ai' can be interpreted as a single vowel with sound AE (as in Husain), or as two vowels AA IH (as in Rai). To perform segmentation, we have a simple program which produces candidates for all possible segments. This program uses a few rules defining the possible consonant and vowel combinations.

Now we simply map these segments to their nearest IL letters (or letter combinations). This is also done using a simple set of mappings, which do not contain any probabilities or contexts. This step generates transliteration candidates. These are then filtered and ranked using fuzzy string matching.

7 Fuzzy String Matching

The initial steps use simpler methods to generate transliteration candidates on the source as well as the target side. They also use no resources on the target (IL) side. The step of fuzzy string matching compensates for the lack of more language specific knowledge during the earlier phase. The transliteration candidates are matched with the words in the target language corpus (actually, words in the word list extracted from the corpus). The fuzzy string matching algorithm we use is finely tuned for Indian Languages and performs much better than language independent approaches like edit distance (Singh et al., 2007). This method can be used for all the languages which use Abugida scripts, e.g. Hindi, Bengali, Telugu, Amharic, Thai etc. It uses characteristics of a writing system for fuzzy search and is able to take care of spelling variation, which is very common in these languages. This method shows an improvement in F-measure of up to 30% over scaled edit distance.

AA	आ|औ	ఆ । ఔ
B	ब	బ
CH	च	చ
D	ड	డ
DH	थ	ధ
F	फ	ఫ
JH	ज	జ
L	ल	ల
M	म	మ
NG	न्ग|न्क	న్గ । న్క
P	प	ప
TH	थ	ధ
UH	ऊ	ఊ
ZH	स|ज	స । జ

Figure 1: Mappings for foreign words. The three columns are for Roman, Devanagari and Telugu

The method for fuzzy string matching is based on the Computational Phonetic Model of Scripts or CPMS (Singh, 2006b), which models scripts (specifically Indic scripts) in terms of phonetic (articulatory) and orthographic features. For calculating the distance between two letters it uses a Stepped Distance Function (SDF). Each letter is represented as a vector of features. Then, to calculate the distance between two strings, it uses an adapted version of the Dynamic Time Warping algorithm (My-

A	अ	అ
AA	आ	ఆ
BH	भ	భ
CH	छ|च	ఛ।చ
D	ड|द	డ।ద
E	ऐ	ఐ
F	फ	ఫ
L	ल	ల
M	म	మ
N	न|ण	న।ణ
OO	ऊ	ఊ
R	र	ర
S	स	స
Z	ज	జ

Figure 2: Mappings for Indian Words

ers, 1980). In the fuzzy string matching method that we use (Singh et al., 2007), an *akshar* (roughly a syllable) is used as the unit, instead of a letter.

8 Discerning Adaptable Transliteration Mechanism (DATM)

We use the above mentioned steps to transliterate a given word based on its origin. In case of ambiguity of word origin both methods are used, and possible transliterations are ranked. Based on the class of the word, the possible pronunciations (for foreign words) and the possible segmentations (for Indian words) are generated. Then, for foreign words, English phonemes are mapped to IL segments. For Indian words, Latin segments are mapped to IL segments.

Now, the transliteration candidates are matched with target language words, using the fuzzy text search method (Singh et al., 2007). Possible transliterations are ranked based on three parameters: word frequency, text search cost and the probability of the word belonging to the class through which it

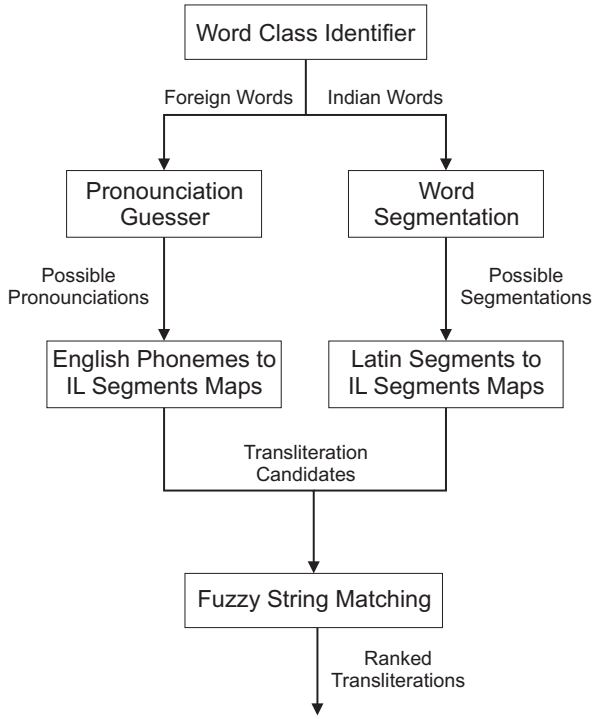

Figure 3: Block Diagram of the Discerning Adaptive Transliteration Method (DATM)

is transliterated. A block diagram describing the method is shown in Figure-3. The ranks are obtained on the basis of a score which is calculated using the following formula:

$$T_t = \frac{log(f_t) * p(C \mid s)}{cost(c,\ t) + K} \quad (1)$$

where T_t is the transliteration score for the target word t, f_t is the frequency of t in the target language corpus, C is the word class (foreign or Indian), s is the source word, c is a transliteration candidate which has been generated depending on the predicted class C, $p(C|s)$ is the probability of the class C given s, $cost(c,\ t)$ is the cost of fuzzy string matching between c and t, and finally K is a constant which determines how much weight is given to the cost of fuzzy string matching.

9 Evaluation

We evaluate our method for two major languages of India: Hindi and Telugu. We compare our results with a very commonly used method (Oh and Choi, 2006) based on bilingual dictionary to learn translit-

Language →	English-Hindi		English-Telugu	
Method ↓	**MRR**	**Pr**	**MRR**	**Pr**
DATM	0.87	80%	0.82	71%
DBL	0.56	47%	0.53	46%
BL	0.43	35%	0.43	37%

DATM: Discerning Adaptive Transliteration Mechanism
DBL: Discerning Baseline Method
BL: Baseline Method

MRR: Mean Reciprocal Rank
Pr: Precision

Table 2: Evaluation on English-Hindi and English-Telugu

erations. As there are no bilingual transliteration dictionaries available for ILs, we had to create our own resources.

9.1 Experimental Setup

We created 2000-word lists which consisted of both foreign and Indian words written in Latin script and their transliterations in Hindi and Telugu. This dictionary was created by people with professional knowledge in both English and the respective Indian language. We only use this list for training the baseline method, as our method does not need training data on the target side. The size of bilingual word lists that we are using is less than those used for experiments by some other researchers. But our approach focuses on developing transliterations for languages with resource scarcity. This setup is more meaningful for languages with scarce resources.

Since, normal transliteration mechanisms do not consider word origin, we train the baseline using the set of 2000 words containing both foreign and Indian words. Alignments from English to respective Indian languages were learned by aligning these lists using GIZA++. The alignments obtained were fed into a maximum entropy classifier with a context window size of 2 (3 is generally considered better window size, but because the training size is not huge, a context window of 3 gave substantially worse results). This method is similar to the grapheme based model as described by Oh and Choi (Oh and Choi, 2006). However, unlike in their approach, the candidate pairs are matched with words in the target language and are ranked based on edit distance (BL).

For our method (DATM), we have used CMU dictionary and a collection of Indian named entities (written in Latin) extracted from web to train the language identification module. We have considered n-grams of order 5 and pruned them by 3500 frequency. In case the foreign word is not found in CMU Speech dictionary, we guess its pronunciation using the method described by Oh and Choi. However, in this case, the context window size is 3.

We also use another method (DBL) to check the validity of our assumptions about word origin. We use the same technique as BL, but in this case we train two models of 1000 words each, foreign and Indian. To disambiguate which model to use, we use the same language identification method as in DATM.

9.2 Results

To evaluate our method we have created word lists of size 200 which were doubly checked by two individuals. These also contain both Indian and Foreign words. We use both precision and mean reciprocal rank (MRR) to evaluate our method against baseline (BL) and discerning baseline (DBL). MRR is a measure commonly used in information retrieval when there is precisely one correct answer (Kandor and Vorhees, 2000). Results can be seen in Table-2. The highest scores were obtained for Hindi using DATM. The MRR in this case was 0.87.

One important fact that comes out from the results is that determining the class of a word and then using an appropriate method can lead to significant increase in performance. This is clear from the results for BL and DBL. The only difference between

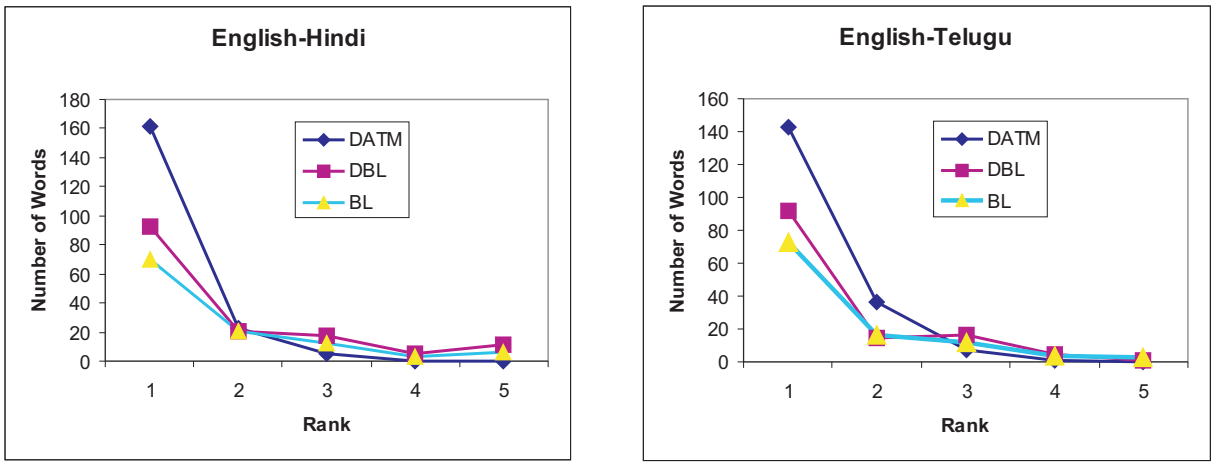

Figure 4: Number of Correct Words vs. Rank. A significantly higher percentage of correct words occur at rank 1 for the DATM method, as compared to BL and DBL methods. This percentage indicates a more practical view of the accuracy transliteration algorithm.

these two was that two different models were trained for the two classes. Then the class of the word was identified (in DBL) and the model trained for that class was used for transliteration.

It should be noted that Yoon et al. (Yoon et al., 2007) have also reported MRR score on Hindi. They have used a number of phonetic and pseudo features, and trained their algorithm on a winnow classifier. They tested their algorithm only for named entities. They have considered a relatively limited number of candidate words on the target language side (1,500) which leads to 150k pairs on which they have evaluated their method. They have reported the results as 0.91 and 0.89 under different test conditions. In case of our evaluation, we do not restrict the candidate words on the target side except that it should be available in the corpus. Because of this formulation, there are over 1000k words for Hindi and over 1800k words from Telugu. This leads to a extremely high number of pairs possible. But such an approach is also necessary as we want our algorithm to be scalable to bigger sizes and also because there are no high quality tools (like named entity recognizers) for Indian languages. This is one of the reason for relatively (compared to figures reported by other researchers) low baseline scores. Despite all these issues, our simpler approach yields similar results.

Figure-4 shows how the number of correct words varies with the rank.

Two possible issues are the out of vocabulary (OOV) words and misspelled or foreign words in the IL corpus. The OOV words are not handled right now by our method, but we plan to extend our method to at least partially take care of such words. The second issue is mostly resolved by our use of fuzzy string matching, although there is scope for improvement.

10 Conclusions and Further Work

We presented a more general and adaptable method for transliteration which is especially suitable for Indian languages. This method first identifies the class (foreign or Indian) of the word on the source side. Based on the class, one of the two methods is used for transliteration. Easily creatable mapping tables and a fuzzy string matching algorithm are then used to get the target word. Our evaluations shows that the method performs substantially better than the two baselines we tested against. The results are better in terms of both MRR (up to 0.44) and precision (45%). Our method is designed to be used for other applications like tolerant input methods for Indian languages and it uses no resources on the target languages side except an unannotated corpus. The results can be further improved if we consider context information too.

We have also shown that disambiguating word origin and applying an appropriate method could be

critical in getting good transliterations. Currently we are assuming that the word to be transliterated is in the target language corpus. We plan to extend the method so that even those words can be transliterated which are not in the target language corpus. We are also working on using this method for building a tolerant input method for Indian languages and on integrating the transliteration mechanism as well as the input method with an open source NLP friendly editor called Sanchay Editor (Singh, 2008).

References

N. AbdulJaleel and L.S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. *Proceedings of the twelfth international conference on Information and knowledge management*, pages 139–146.

N. Aswani and R. Gaizauskas. 2005. A hybrid approach to align sentences and words in English-Hindi parallel corpora. *Proceedings of the ACL Workshop on" Building and Exploiting Parallel Texts.*

M.W. Davis and W.C. Ogden. 1998. Free resources and advanced alignment for cross-language text retrieval. *Proceedings of the 6th Text Retrieval Conference (TREC-6)*, pages 385–402.

M. Ganapathiraju, M. Balakrishnan, N. Balakrishnan, and R. Reddy. 2005. OM: One Tool for Many (Indian) Languages. *ICUDL: International Conference on Universal Digital Library, Hangzhou*.

L. Larkey, N. AbdulJaleel, and M. Connell. 2003. What's in a Name? Proper Names in Arabic Cross-Language Information Retrieval. Technical report, CIIR Technical Report, IR-278.

A. Llitjos and A. Black. 2001. Knowledge of language origin improves pronunciation of proper names. *Proceedings of EuroSpeech-01*, pages 1919–1922.

M.G.A. Malik. 2006. Punjabi Machine Transliteration. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 1137–1144.

J. May, A. Brunstein, P. Natarajan, and R. Weischedel. 2004. Surprise! What's in a Cebuano or Hindi Name? *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):169–180.

C. S. Myers. 1980. *A Comparative Performance Study of Several Dynamic Time Warping Algorithms for Speech Recognition*. Ph.D. thesis, M.I.T., Cambridge, MA, Feb. http://gate.ac.uk.

J.H. Oh and K.S. Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7.

J.H. Oh and K.S. Choi. 2006. An ensemble of transliteration models for information retrieval. *Information Processing and Management: an International Journal*, 42(4):980–1002.

A. Rathod and A. Joshi. 2002. A Dynamic Text Input scheme for phonetic scripts like Devanagari. *Proceedings of Development by Design (DYD)*.

Anil Kumar Singh, Harshit Surana, and Karthik Gali. 2007. More accurate fuzzy text search for languages using abugida scripts. In *Proceedings of ACM SIGIR Workshop on Improving Web Retrieval for Non-English Queries*, Amsterdam, Netherlands.

Anil Kumar Singh. 2006a. Study of some distance measures for language and encoding identification. In *Proceedings of ACL 2006 Workshop on Linguistic Distance*, Sydney, Australia.

Anil Kumar Singh. 2006b. A computational phonetic model for indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands.

Anil Kumar Singh. 2008. A mechanism to provide language-encoding support and an nlp friendly editor. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, India.

RMK Sinha. 2001. Dealing with unknowns in machine translation. *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, 2.

S.Y. Yoon, K.Y. Kim, and R. Sproat. 2007. Multilingual Transliteration Using Feature based Phonetic Method. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 112–119.