Transliteration as Alignment vs. Transliteration as Generation for Crosslingual Information Retrieval

Anil Kumar Singh* — **Sethuramalingam Subramaniam*** — **Taraka Rama***

* *Language Technologies Research Centre, IIIT, Hyderabad, India*
{anil@research, sethu@research, taraka@students}.iiit.ac.in

ABSTRACT. *Crosslingual Information Retrieval (CLIR) usually requires query translation and, due to named entities in the case of IR, query translation requires a good transliteration system when writing systems differ. Transliteration can be seen as a problem of generation or alignment. For IR, since we can extract a word list from the corpus being searched, it should be seen as an alignment problem. The shift from generation to alignment can lead to higher transliteration accuracies and significant improvements in the CLIR results. We were able to achieve an increase (over generation) in the CLIR Mean Average Precision by 22.66% and 29.08% for English to Hindi and English to Marathi, respectively.*

RÉSUMÉ. *La recherche d'information interlingue implique la traduction des requêtes. En raison du grand nombre d'entités nommées dans les requêtes, des systèmes de translittération efficaces doivent être mis en œuvre quand les systèmes d'écriture diffèrent. Comme l'extraction de liste de mots cibles à partir des corpus interrogés est possible, nous préférons assimiler la translittération à un problème d'alignement plutôt qu'à un problème de génération. Ce choix conduit à de meilleures translittérations et à des améliorations importantes des réponses aux requêtes. Nous avons ainsi amélioré la précision moyenne de notre système de 22,66 % de l'anglais vers l'hindi et de 29,08 % de l'anglais vers le marathi.*

KEYWORDS: *Multilingual, Transliteration, Crosslingual Information Retrieval, Surface Similarity, Generation, Alignment, Indian Languages, English, Hindi, Marathi.*

MOTS-CLÉS : *multilinguisme, translittération, recherche d'information interlingue, génération, alignement, anglais, hindi, marathi.*

1. Introduction

With the coming together of all kinds of data in different languages on the Web and elsewhere, the search for better methods for multilingual and crosslingual text processing for applications like Information Retrieval (IR) has gained a lot of attention. However, for some languages which do not even have the basic language resources such as dictionaries (monolingual or multilingual) that are good enough to be used for practical applications, we need to explore methods which can perform reasonably well even in the absence of such resources.

The above is true for most of the major Indian languages, almost all of them spoken by more than a hundred million people. Efforts to develop robust methods specifically for these languages have only recently gained momentum. One of the important applications on which work is going on is Crosslingual Information Retrieval (CLIR), and one of the major problems for this application is query translation since CLIR systems are usually based on query translation. Since queries are usually not grammatical sentences, we cannot effectively use Machine Translation (MT) systems for this purpose. In any case, good MT systems for these languages are not available. In this situation, good transliteration is essential for improving query translation. It has special relevance for the present case (English to Indian languages) where the transliteration module can be as important or perhaps even more important than the rest of the query translation system. It has been shown that just good transliteration can help in getting tolerable performance for CLIR systems. We can, of course, use many other techniques and heuristics to add to transliteration, but highly accurate transliteration methods are still a key to good CLIR for Indian languages, apart from being very useful for other Natural Language Processing (NLP) applications such as Machine Translation (MT).

As Surana and Singh (2008) (among others) have pointed out, one of the main reasons for the importance of transliteration from the NLP point of view is that the Out Of Vocabulary (OOV) words are quite common since all lexical resources are very limited in practical terms. These OOV "words" include named entities, technical terms, rarely used or "difficult" words and words borrowed from other languages. From the IR perspective, transliteration becomes crucial because, as pointed by Cheng *et al*. (2004), the OOV words seem to constitute 83% of the highly frequent query terms. Analysis conducted by Davis and Ogden (1997) revealed that around 50% of the OOV words in the queries were named entities. Moreover, it has been proved that CLIR performance (in terms of Mean Average Precision) was reduced by more than 50% when named entities in the queries were not transliterated (Abdul Jaleel and Larkey, 2003). Intuitively also it seems logical that having even just one named entity in the query can make retrieval more easy and more accurate as it reduces the search space drastically and is one of the most reliable bases for estimating the relevance of the document. Named entities have to be transliterated for language pairs that use different writing systems.

In a multilingual scenario, the role of transliteration is even more important for the same reason and due to the fact that OOV words cannot be translated using a multilingual dictionary. Transliteration becomes vital in translating the OOV words across languages.

We propose that transliteration should be seen as two related but different problems: **transliteration as generation** and **transliteration as alignment**. The latter would usually include the former. Alignment assumes that a target list is available. This may sound like an unreasonable assumption, especially for resource-scarce languages, but it is just a statement of fact in the case of IR, where we really do have the corpus in which we are searching something and from this corpus we can easily extract the target list. Thus, for CLIR, transliteration should be seen as a problem of **transliteration as alignment**. As we show later in the article, it is possible to get much higher accuracies for alignment than for only generation. We present the results of transliteration experiments with one method based on generation alone, two variants of a method based on alignment that uses simple techniques for generation, another method based on alignment that also uses simple techniques for generation and two variants of a method based on alignment that uses a sophisticated method for generation. For the two variants of the last method, we were able to achieve an accuracy of 80% and 84% on the ACL NEWS Shared Task data for transliteration [1], which is quite high for the case of English to Hindi. Although this accuracy is achieved by mixing up the training, development and test data to get the target list (not for training the generation system), it is a valid way to evaluate for the case of alignment since here we are interested more in CLIR and less, say, in Machine Translation. Also, we focus mainly on transliteration as used for CLIR, rather than on improving the overall performance of the CLIR system. For CLIR, we were able to increase the Mean Average Precision for alignment (over generation) by 22.66% and 29.08% for English to Hindi and English to Marathi, respectively.

The article is organized as follows. In the next section we briefly mention the related work on transliteration. In Section 3, we discuss the case of English to Indian language transliteration and its special problems. Section 4 describes the methods of transliteration we use for our experiments. In Section 5, we present the results of the experiments on transliteration alone. In Section 6, we move on to CLIR and describe the CLIR system used. Section 7 briefly describes the CLIR experimental setup and Section 8 presents the results of these experiments. Finally, we conclude in Section 9.

2. Related Work on Transliteration

Transliteration of words from a source language to a target language is mostly considered to be a problem of generation of target language equivalents. Most of the methods used for this purpose are based on statistical approaches. As transliteration is not a new problem, even though for Indian languages the systems still lag behind,

1. http://www.acl-ijcnlp-2009.org/workshops/NEWS2009

there has been a lot of work on this problem. The major techniques for transliteration can be broadly classified into two categories, viz grapheme-based and phoneme-based approaches. Knight *et al*. (1997) developed a phoneme-based, statistical model using finite state transducer that performed back-transliteration using transformation rules. Paola and Khudanpur (2003) used another phoneme-based approach using transformation based learning algorithm. Yaser and Knight (2002) used a grapheme-based approach that maps English letter sequences to Arabic letters. Abdul Jaleel and Larkey (2003) demonstrated a simple, statistical technique for building an English-Arabic transliteration model using Hidden Markov Model (HMM) and alignments obtained from GIZA++ (Och and Ney, 2003) [2]. Goto *et al*. (2003) used a maximum entropy based model for English-Japanese transliteration which predicts the Japanese equivalents for English chunks using their contextual information probabilities. Li *et al*. (2004) presented a joint source-channel n-gram model for English-Chinese transliteration using orthographic alignments obtained from an English-Chinese bilingual dictionary.

In the context of Indian languages, Aswani and Gaizauskas (2005) have used a transliteration similarity based technique 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 methods can only be appropriate for aligning parallel texts as the number of possible candidates is quite small. This work is implicitly based on transliteration as alignment, but it is different from the alignment-based approach that we present in this article and it does not explore the idea beyond alignment of parallel corpus. Malik (2006) proposed a rule-based method of transliterating Punjabi language words from Shahmukhi (Arabic script) to Gurumukhi script (derivation of Landa, Shardha and Takri, some old scripts of the Indian subcontinent). Ekbal *et al*. (2006) used a modified source-channel model for Bengali-English machine transliteration. Their work is an extension of the earlier works (Goto *et al.*, 2003; Li *et al.*, 2004). They also used language specific heuristics for identifying transliteration units in Bengali. Surya *et al*. (2008) presented a discriminative model using Conditional Random Fields (CRF) and HMM-alignments for Hindi-English transliteration. This work can be compared to the earlier work by Abdul Jaleel and Larkey (2003) where they used a generative model (HMM). Srini *et al*. (2008) proposed a mapping-based Compressed Word Format (CWF) algorithm for English-Tamil transliteration. They compared their system performance with the discriminative model proposed by Surya *et al*. (2008).

One of the methods presented in the current work is an extension of the transliteration system proposed by Surana and Singh (2008) for English to Hindi transliteration. They proposed a word origin based method of transliteration in which possible target equivalents are generated based on whether the given word is of Indian origin or foreign origin. Fuzzy string matching (Singh *et al.*, 2007) to estimate the surface

2. `http://www.fjoch.com/GIZA++.html`

similarity [3] is used for aligning the generated equivalents and arriving at the correct target equivalent. This approach does not require a "parallel corpus" for generating and ranking the candidates, but it does need the target side word list for fuzzy string matching. From the IR perspective, our work is an extension of the work by Subramaniam *et al*. (2009). They also used transliteration based on estimation of surface similarity for CLIR and were able to achieve reasonably good performance. However, their experiments were on the relatively easier task of Indian language to Indian language CLIR.

Coming to the current work, we consider the much harder task of English to Indian language transliteration and its application to CLIR. Also, for our final experiment, we combine a state-of-the-art generation method (Statistical Machine Translation) with a state-of-the-art alignment technique (fuzzy string matching based on a Computational Phonetic Model of Scripts). In the next section we will discuss the special problems one has to face in this case.

3. The Case of English to Indian Language Transliteration

Transliteration from English to Indian languages is basically similar to the problem of phonetic transcription because the scripts used for Indian languages are highly phonetic in nature and there is a close correspondence between letters and phonemes.

There are several reasons why the problem of English to Indian language transliteration has been given more urgency than Indian language to English or Indian language to Indian language. One reason is that most Indians using computers also know English to some degree. Another reason is that, till recently, computers had hardly any support for Indian languages and most of the Indians using computers were not (in fact, still are not) able to type text directly in Indian languages. Therefore, if they want to search for some text in Indian languages, they would still find it easier to type their queries in the Latin script, rather than some Indian script, even if input for that script is enabled on their computer. The reason more relevant here is that we need a good transliteration system for searching text in a language different from that in which the query is entered, the usual CLIR scenario. Even if the query is in an Indian language, the Indian users are more likely to type the query using the Latin alphabet, but they might want to search for documents in Indian languages, thus effectively turning (in such a case) the problem of IR into CLIR.

Given this background, the two major problems for English to Indian language transliteration are:

1) *Source side ambiguity*: the relatively irregular spelling of English (in phonetic terms) and the lack of a commonly accepted Roman notation for typing Indian language text;

3. Roughly speaking, orthographic and phonetic similarity, but it could be applied to other linguistic modalities, such as sign language.

2) *Target side ambiguity*: a high degree of variation and non-standardization of spellings in Indian languages (in their respective scripts).

These two problems make the task much harder at both the source as well as at the target side. The relative irregularity of spellings at the source side means that the number of possible transliteration candidates that can be generated is very large for even a medium-sized word. The high variation at the target side means that it is very difficult to decide (often for even humans) whether a generated candidate is an acceptable word (or name), even if we ignore the role of the context.

It needs to be emphasized that the two problems will effectively still be present when we try to transliterate "romanized" Indian languages text (say, IR queries) to Indian language scripts. This is because there is no popular commonly accepted notation (although there are innumerable ones for academic purposes) for typing Indian language text in the Latin script. Since the mappings from Latin letters to Indian script letters are not only many to many but highly ambiguous, the irregular spelling aspect of the problem actually becomes more important in this case.

To illustrate ambiguity on the source side, as Surana and Singh (2008) mentioned, a word like "nOkarI" (job) can be written in Roman at least as *naukri*, *nokri*, *naukari*, *naukary*, *nokari*, *naukarii* and *naukaree*, with all of them getting a large number of hits on Google (Table 1). For the target side, we give an example (Figure 1) consisting of words (or, more accurately, tokens) of four kinds to show the level of ambiguity that an English to Indian language transliteration system has to deal with. Note that these examples (on the source as well as the target side) are not exceptionally difficult ones. It is true that there are simpler cases, but such difficult cases are very common. Also, names are usually harder to transliterate than words.

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. *Source side ambiguity for English to Indian languages transliteration. Variations of a Hindi word nOkarI (job). The numbers are pages returned when searching on Google.*

On the positive side, there are a lot of similarities among the writing systems used for Indian languages (Singh, 2006) and this fact make it much easier to build systems which can transliterate from English to many Indian languages using the same method and the same basic setup. However, to achieve this, one has to design the system in such a way that it benefits from all the similarities among the scripts. The methods that we consider in this article are scalable and robust in this sense.

Type	Word	Acceptable transliterations
English word	Azure	अज्योर, अज्यॉर, अज़्योर, अज़्यॉर, एज्योर, एज्यॉर, एज़्योर, एज़यॉर, अज्यौर, अज़्यौर, एज्यौर, एज़्यौर
Foreign name	Norfolk	नोरफोल्क, नोरफ़ोल्क, नोरफॉल्क, नोरफ़ॉल्क, नोरफोक, नोरफ़ोक, नोरफॉक, नोरफ़ॉक, नॉरफोल्क, नॉरफ़ोल्क, नॉरफॉल्क, नॉरफ़ॉल्क, ...
Indian language word	Raja	राजा, राज, राज, रजा, राज़, रज़ा, रज
Indian name	Madhulika	माधुलिक, मधुलिक, माधुलिका, माधुलिका, मधुलीक, माधुलीक, ...

Figure 1. *Target side ambiguity for English to Indian languages transliteration for four different types of words. The third columns shows some of the possible transliterations, all of which could be acceptable. For English words and foreign names (rows 1 and 2), all the variants given are acceptable phonetically, partly due to lack of standard conventions about transliterating English words and foreign names and partly because the names are supposed to be transliterated phonetically and the pronunciation might vary with, say, the accent with which English is spoken by the writer. For Indian language words (row 3), there is ambiguity because all the transliterations given are valid words. For Indian names (row 4), the ambiguity is due both to the huge variety in possible names and to the fact that people spell their names differently.*

In the light of the above, there are some difficult questions that a researcher or a developer working on transliteration for English to Indian languages has to ask. Some of them are:

– How to take care of the source side ambiguity?

– How to resolve the target side ambiguity?

– What kind of method is better for generating candidates?

– On what basis should the candidates be pruned?

– How to rank and select the candidates?

– How to evaluate the system?

While discussing these questions, we will assume that all transliteration methods work according to the following general conceptual scenario, even if they do not follow each step explicitly:

1) generate segmentations of the source word (or token);

2) prune segmentations;

3) generate transliteration candidates;

4) prune transliteration candidates;

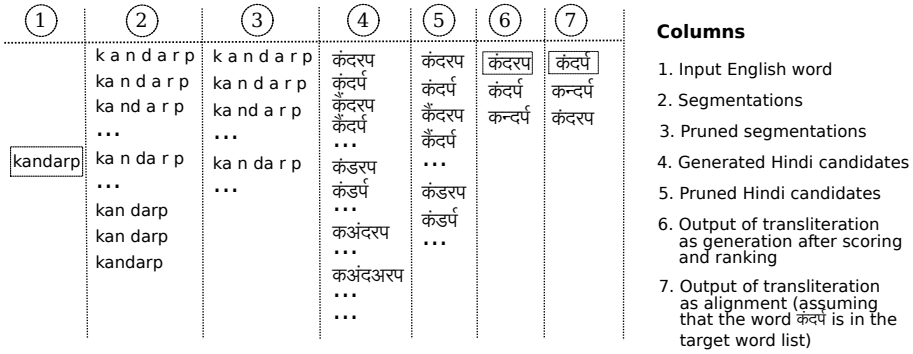

Figure 2. *An example showing the usual steps in transliteration. The last two columns show the returned candidates for transliteration as generation and transliteration as alignment, respectively.*

5) assign scores to transliteration candidates;

6) rank and prune the remaining transliteration candidates;

7) return one or more transliterations.

Figure 2 shows an example of the above steps (except that of scoring the candidates). The last two columns in the figure show the returned candidates for transliteration as generation and transliteration as alignment, respectively.

Pruning segmentations as well as candidates is required because for certain language pairs (such as English to Indian languages), the number of transliteration candidates can otherwise become so large that it is not computationally feasible to use them all, even if the CLIR system is going to ignore most of the invalid ones. But the bigger problem is that the more candidates there are, the greater the chances that some of them will (wrongly) get aligned to some word in the target word list.

An example of works where the transliteration method explicitly includes almost all of the above steps is by Qu *et al*. (2003) and it was also used for CLIR. This work (on English to Japanese transliteration) is also an example of using transliteration as alignment, rather than transliteration as generation. The alignment (or "validation") was performed in their case against a word list (or dictionary) extracted from a monolingual Japanese corpus. In a related work on the same problem, Qu and Grefenstette (2004) also took the origin of the name into account (using language identification techniques) and they tried to use the Web as the corpus for alignment or validation. The Web (and specially Wikipedia) is, in fact, being increasingly used for overcoming the lack of resources, including transliteration dictionaries (Sato, 2009). For Indian languages, the problem is that Wikipedia, and Web in general, does not yet have as much content to mine named entities and other OOV words as for languages like English.

Going back to the points mentioned above, to take care of ambiguity on the source side, (ideally) we have to make sure that all the valid candidates are generated. This can perhaps best be done by a statistical technique, but such a technique requires parallel data (list of source words and their correct transliterations). Some pruning of segmentations and candidates is possible by simple heuristics. These heuristics, in the case of Indian languages, can be based on the fact that the scripts follow some rules about which kinds of characters can follow which kinds of other characters. For example, a vowel cannot be followed by a vowel sign (*maatraa*) and a vowel modifier or a consonant modifier cannot follow another or each other. We use a simple grammar (Singh *et al.*, 2007) to prune candidates containing such impossible sequences. Another heuristic is to specify a maximum length for segments, e.g. for English to Indian languages, one can set a maximum segment length of 3 or 4, because longer sequences cannot represent a single letter in the Indian language script. However, for more sophisticated pruning, we again need parallel data and some techniques like the ones used for Statistical Machine Translation (SMT). But if we use a method like the one used by Surana and Singh (2008), then a simple technique based on rough mappings may be good enough for the source side ambiguity because fuzzy string matching (estimation of surface similarity) or "validation" at the target side could ensure that the right candidate is selected. But if we do not assume that a list of all the target side words is available, then pruning and ranking on the target side does need statistical techniques.

It can be mentioned here that the statistical techniques are mainly of two kinds. The first are the easy to implement ones but less effective. In this category come letter based n-gram language models. These models can be used both for pruning as well as for ranking, but when used alone they are hardly sufficient. Still, they can be used intelligently both on the source as well as the target side. The second kind of techniques are those which use parallel data. They can be very effective, but they need hard-to-prepare resources and are only as effective as the resources they use. One of the interesting areas of research is how the statistical techniques can be combined with other techniques to ensure that the system can work with low quality and small size resources or even without them to some extent. The methods that we experimented on are an exploration in this direction.

One important but less addressed issue is that of evaluation. As the examples given earlier show, the high level of variation at the target side makes it difficult to properly evaluate a technique. For example, the reference data provided for the ACL Named Entities Workshop [4] for the transliteration task (English to Hindi) has only one correct transliteration in most cases, whereas the correct (or acceptable) transliterations are almost always more than one. For the words given in the test data for this shared task, it would be very common to have four or more acceptable transliterations. This means that the evaluation using this data is going to be too strict. However, it is very hard to prepare data that lists all possible transliterations. There might even be argument

4. `http://www.acl-ijcnlp-2009.org/workshops/NEWS2009`

about whether something is acceptable or not. As Figure 1 shows, it may not even be practical to list all the acceptable transliterations.

The solution that we suggest is that evaluation for English to Hindi transliteration should be based on fuzzy string matching, instead of exact matching. The rank along with the fuzzy string matching score can be used to design a better metric for this task. However, not just any approximate string matching would serve the purpose. The technique selected for fuzzy string matching should be able to estimate surface similarity (i.e., phonetic and orthographic similarity in the present case) because, for example, p and f are less distant than p and r. The usual edit distance methods are not able to account for this. Something like the Computational Phonetic Model of Scripts or CPMS (Singh, 2006) can be used for this purpose [5]. But we leave the design of a better metric as work for the future.

The above discussion is about transliteration in general. However, if we assume that a target list of all correct transliterations can be available, many of the problems mentioned above do not apply. CLIR is a problem where this assumption is valid. Therefore, there is no need to rely only on generation to give the correct transliteration. Instead, we should make the best possible use of alignment. Alignment based on CPMS-based fuzzy string matching can also account for the target side spelling variation.

4. Experiments in Transliteration

We tried six methods for our experiments. The first method uses simple mappings for generation, but uses two variants of the CPMS-based alignment or Fuzzy String Matching (FSM) technique (Singh, 2006; Singh *et al.*, 2007). The first variant (DATM) uses *akshars* (roughly syllables) as the smallest units (Surana and Singh, 2008), while the second (LDATM) uses letters as the smallest units. The second method is a modified version of the DATM method (Naive SMT+FSM) that uses simple mappings for generation, a simple n-gram based method for initial ranking of the candidates and FSM for alignement. It can be seen as a combination of naive SMT and FSM. The third method (SMT) is based on the standard generative Statistical Machine Translation. This method is based on generation alone as there is no alignment against the target list. Finally we use a method (SMT+FSM) that uses SMT for generation and initial ranking of candidates. The remaining candidates are then aligned using the two variants of the CPMS-based technique (AFSM and LFSM). We first present a summary of the FSM alignment technique in the next sub-section. Then we explain all the methods in the following sub-sections.

5. The CPMS-based estimation of surface similarity can be seen at one level as a generalization of edit distance based methods. However, it is different from other such generalizations in the way it is modeled and implemented and the purposes for which it can be used.

4.1. *Fuzzy String Matching (FSM)*

Fuzzy String Matching as used by Singh *et al*. (2007) is based on the idea that written text contains not just orthographic but phonetic information too. We can use all this orthographic and phonetic information to estimate the *surface similarity* of two strings or words. This method is a natural extension of the edit distance based methods except that the cost of substitution is directly related to orthographic and (more importantly) phonetic similarity. Such a method can take into account the fact that /b/ and /p/ are more similar than /p/ and /t/.

The CPMS is used to estimate surface similarity. This is a model of scripts [6] that uses the characteristics of scripts such that the phonetic information available in written text is effectively utilized to estimate similarity. Each letter is represented as a vector of features. The model also consists of an important component called the Stepped Distance Function (SDF) that gives the orthographic and phonetic similarity of two letters (Figure 3). A dynamic programming alignment algorithm such as Dynamic Time Warping (DTW) can then be used to align two strings and get their similarity. DTW is used just as a way to calculate the edit distance, using the phonetic features of letters. It could be replaced by some other alignment method, without affecting the core CPMS.

Fuzzy string matching can be expressed as estimation and maximization of surface similarity (Singh *et al.*, 2007):

$$S_s \;=\; f(w_1, w_2, A, W, W_n, P, P_n, D) \qquad [1]$$

where f is a function representing an alignment algorithm (such as a dynamic programming algorithm), w_1 and w_2 are the words or strings being compared, A is the alphabet, W is the set of orthographic features, P is the set of phonetic features, W_n and P_n are the sets of numerical values assigned to the orthographic and phonetic features, and D is a distance function for calculating the similarity between two letters.

D can itself be defined as:

$$D \;=\; f(l_1, l_2, A, W, W_n, P, P_n) \qquad [2]$$

where l_1 and l_2 are the two letters being compared as part of the alignment algorithm. The distance function is a stepped distance function (Figure 3) that gives the scores for phonetic similarity of two letters based on a hierarchy of phonetic features. Features at different levels in the hierarchy have different weights.

In the case of *akshar*-based FSM (AFSM), a list of all possible *akshars* (less than 2000 in all or most languages) is created first. Then the similarity of every pair of *ak-*

6. Implemented so far for Brahmi origin scripts, which are used by most of the major Indian languages.

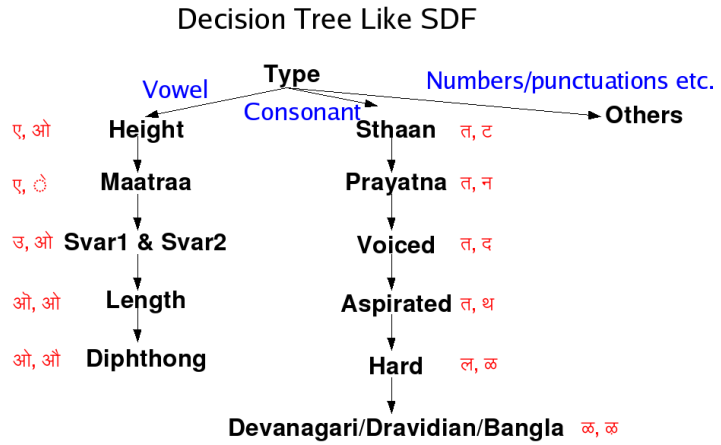

Figure 3. *Stepped distance function: various steps differentiate between different kinds of letters. At the end, a quantitative estimate of the orthographic and phonetic distance is obtained.*

shars is calculated using the SDF and DTW. These *akshar* pair similarities are stored. To get the similarity of strings now, we just need to run the DTW alignment over *akshars* such that the cost of substitution is obtained from the stored *akshar* pair similarity table.

4.2. *Discerning and Adaptable Transliteration Mechanism (DATM)*

This was the method used by Surana and Singh (2008). We use it as one of the baselines, the other being the generation-alone standard SMT method. The basic logic on which this system is based is shown in Figure 4. The first idea is that the way a word can be transliterated depends on the origin of the word (Oh and Choi, 2002; May *et al.*, 2004), i.e., the method of transliteration may need to be different for words of different origin. Llitjos and Black (2001) had shown that the knowledge of language origin can substantially improve pronunciation generation accuracy. They used probabilities of all trigrams to belong to a particular language as a measure to disambiguate word origin. We, like Surana and Singh, used a more accurate method that has been successfully used for language and encoding identification (Singh, 2006a).

The second idea is that if we have the list of target words, we can use fuzzy string matching to select the right candidate(s), even if the list of generated candidates does not include the right candidate(s). The method for fuzzy string matching is AFSM as explained above.

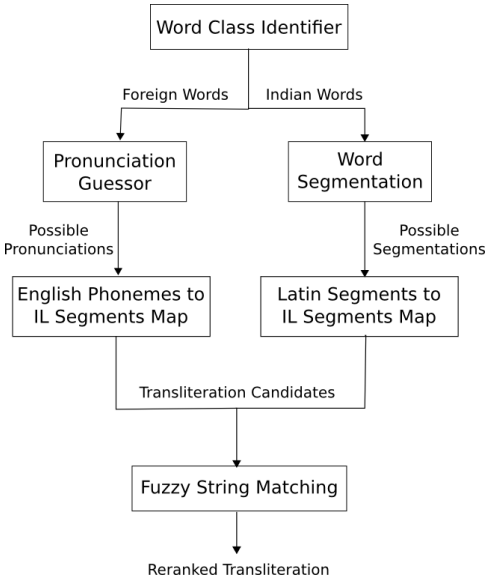

Figure 4. *Block diagram of the Discerning Adaptive Transliteration Method (DATM)*

Depending on the word origin, candidates are generated differently. Two main classes of words are assumed: Indian and Foreign (not to be taken literally). For the "Foreign" words, the CMU speech dictionary is used as a look-up and if a word is not present in this dictionary, some simple mappings are used to segment and generate candidates. For "Indian" words, different but equally simple mappings are used. Some simple heuristics are also applied. For example, the consonant modifier *nukta* (a kind of diacritic that converts one consonant symbol to another phonetically close consonant, usually one that come into the language due to loanwords from English, Persian etc) is ignored for alignment purposes because for almost all the words involving the *nukta*, the spelling variant without it is also considered acceptable. The important thing is that no expert linguistic knowledge is used either for preparing mappings or for applying heuristics and no costly resource is used. The method can work even without the speech dictionary. However, the main drawback of this method is that it assumes that a target list of all words is available. This is a restrictive assumption, but it can be used for CLIR.

We also use a variant (LDATM) of this method that uses LFSM for alignment.

4.3. *Naive SMT with Fuzzy String Matching (FSM)*

This method uses a simple approach for generation, which is, in principle, similar to SMT but uses naive techniques. To implement it, we used a modified version of

the DATM implementation. The major modifications are as follows. We don't use the word origin guessing component because word origin guessing causes errors in many cases. When it works, it can be very effective, but when it goes wrong, it is very difficult to recover from the error. Instead, we generate candidates assuming both classes, i.e., using both ("Indian" and "Foreign") methods. We first merge these candidates together. Then we use a letter based n-gram model to get the sequence probabilities of candidates (normalized for the length of the sequence). The candidates are ranked based on these probabilities as the scores. Finally, top N of these are given to the fuzzy string matching component to return the right candidate. In cases where no (or not enough) candidates are returned, the best candidates based on sequence probabilities are included in the output. This allows us to give an output even if the word is not present in the target word list or could not be matched due to the settings of the fuzzy string matching component. This is a major advantage over the DATM method.

4.4. *Transliteration as Statistical Machine Translation (SMT)*

The next method for which we present the results is simply using the available tools for SMT. GIZA++ [7] is used for generating letter alignments from the parallel data provided for the ACL 2009 NEWS shared task. Then Moses [8] is used for generating phrase transliteration tables and also for decoding. The output of the decoder is the output of the transliteration system.

4.5. *SMT with Fuzzy String Matching (FSM)*

In this method, we took the output of the SMT-based system and applied fuzzy string matching with the target word list. Like in the Naive SMT+FSM method, if nothing is returned from the fuzzy string matching (FSM) module, we include the best candidates from the SMT output. And like in the case of DATM, we tried both AFSM and LSFM for alignment, giving two variants of the method.

5. Evaluation and Results for Transliteration

For evaluation of the transliteration system alone (i.e., before using it for CLIR), we used the data provided for the ACL NEWS shared task. The evaluation metric used are also the same as those used for the shared task, namely accuracy (top 1), Mean Reciprocal Rank (MRR) and Mean F-score. Since we are interested in transliteration as alignment for the purpose of CLIR, we need a target list for the methods based on alignment (i.e., all except SMT). We created this list by merging the training,

7. http://www.fjoch.com/GIZA++.html
8. http://www.statmt.org/moses

Language →	English-Hindi		
Method ↓	**Accuracy**	**MRR**	**Mean F-score**
DATM^A	50.10%	51.32%	60.77%
LDATM^A	68.30%	73.34%	83.55%
Naive SMT+AFSM^A	56.60%	58.32%	82.53%
SMT^G	46.30%	57.33%	86.60%
SMT+AFSM^A	80.70%	81.34%	92.82%
SMT+LFSM^A	84.50%	88.94%	93.96%

Superscript G: Generation
Superscript A: Alignment

AFSM: Akshar-Based Fuzzy String Matching
LFSM: Letter-Based Fuzzy String Matching
DATM: Discerning Adaptive Transliteration Mechanism
LDATM: DATM with LFSM for alignement, instead of AFSM
Naive SMT: Candidates generated from mappings, ranked using language model
SMT: Transliteration as Statistical Machine Translation

Table 2. *Evaluation of English to Hindi transliteration as generation^G and as alignment^A*

development and test data provided for the ACL shared task. Naturally, the results in this case do not apply for the transliteration as generation case.

The following observations can be made from Table 2. Both DATM and Naive SMT+FSM perform better than SMT if the correct transliterations are in the target word list. It is not a trivial task even if the correct transliterations are present in the target word list because the source and the target side ambiguities are still there.

The kinds of errors encountered are different for different methods. For DATM, the largest number of errors seem to be due to wrong identification of the word origin. This happens most frequently with words of smaller lengths (less than seven characters). Short words also sometimes tend to get wrongly matched during alignment. There is another category of errors for DATM that seems to be purely due to the way AFSM is implemented, i.e., using the SDF-based similarity in an indirect way. Such errors do not occur with LDATM or SMT+LSFM.

The errors for SMT are mainly due to two reasons. The first is the limitation of the parallel corpus from which the system is trained. The second is the fact that SMT is just not enough to produce exactly the right candidate as it relies heavily on the language model, which has no way to account for the target side ambiguity. This is a general problem for all methods based on generation alone, unless word-context information is also used.

Theoretically, it might appear at first sight that if all the possible candidates are generated, then a good alignment-based method should be able to reach accuracies

very near to 100% since all the correct transliterations are present in the target list. However, apart from the source and target side ambiguities, there are other practical limitations to this. One is, as noted above, the limitation of the parallel corpus for generation (if SMT is used for generation) or the limitation of mappings (if mappings are used). Another is the fact that for aligning against a very large list (several hundred thousand entries) we have to use an optimized method, which can sometimes miss the correct entry.

To summarize, if a good target list is available, SMT+FSM is the best option, but Naive SMT+FSM (like DATM) has the advantage that it does not need any parallel data to learn from. If a good target list is not available, then SMT is a good option, though it might be interesting to explore how much Naive SMT+FSM can be improved and whether it can be made to give results comparable to SMT, without using a target list. The obvious limitation with fuzzy string matching is that it can only improve SMT if a good target list is available. But this is not a problem in the CLIR case.

It may be pointed out that all the results are very much on the lower side (as compared to those reported by Surana and Singh) because of the way in which evaluation is performed. In most cases, only one transliteration is considered to be the correct one, though others could have been acceptable.

For CLIR, we used the above methods of transliteraton to help "translate" the queries. In the following sections we will describe the CLIR system used and its evaluation.

6. CLIR System Architecture

Our CLIR system is based on a "dictionary-based" method of query translation. Dictionaries gathered from different sources were also used for this purpose, though as indicated in the beginning, they are very inadequate. Named entities found in the queries are identified and transliterated using the transliteration methods described earlier. Documents are indexed using the Lucene [9] framework and a vector-based model is used for ranking the documents. Lucene's OKAPI BM25 is used as the similarity metric for scoring the documents.

6.1. *Query Processing*

The query processing module consists of an n-gram based translation of query words using bilingual lexicons, identification of named entities using named entity recognizers, transliteration of the identified named entities and a boolean query scoring sub-module. A weighted Boolean query is generated as the output from the query processing module.

9. `http://lucene.apache.org/`

6.1.1. *Query Translation*

Query translation primarily involves translation of query words using bilingual lexicons. An n-gram based approach was used to match the entries that have two or more words in the lexicon, thereby increasing the probability of translating multi-word entries present in the query. The details of the dictionaries used in our experiments are given in Table 3.

Language Pair	**Dictionary Size (Number of Entries)**
English-Hindi	22059
English-Marathi	7802

Table 3. *Dictionary statistics*

6.1.2. *Named Entities Identification*

Identification of named entities or the Out Of Vocabulary words (OOVs) in the given query is very critical in deciding upon which are the words to be transliterated and which are not. Such a binary classification is more relevant for this purpose than recognizing the class of the named entities. For English queries, we used the Named Entity Recognizer (NER) from the Stanford NLP Group [10] to identify the named entities present in the queries. The identified named entities in the queries are passed to the transliteration module for transliteration.

6.1.3. *Query Scoring*

Once the source language queries are translated and transliterated, the resultant target language keywords are used to construct Boolean queries using the OR operator. We use different scores for the words originated from the different parts of the source topic. Let W_t be the weight assigned to target language words originating from the title section of the source topic. Let W_d be the weight assigned to target language words originating from the description section of the source topic. And let W_n be the weight assigned to target language words originating from the narrative section of the source topic. Then the ordering of weights can be given as:

$$W_t > W_d > W_n$$

If a particular keyword occurs in multiple sections of the query, it has to be given a greater score compared to the other keywords. Hence, the cumulative weight for each word is calculated based on the number of occurrences. Other important keywords like years, numbers, etc are also given higher weight factors. If t_i be the translated query word in the Boolean query and w_i be the scoring weight associated to it, then the final query output T for a given source language query from the system would be of the form:

10. `http://nlp.stanford.edu/ner/index.shtml`

```
<top lang="en">
<num>27</num>
<title>Relation between India and China</title>
<desc>India and China's bi-lateral relation in terms of economy,
diplomacy, science, technology and civil-aviation.</desc>
<narr>Information about the relationship between India and China
with regard to economy, diplomacy, science, technology and trade
is relevant.</narr>
</top>
```

Figure 5. *A sample test topic in English*

$$T = \bigcup_i w_i.t_i \quad [3]$$

In the next section we briefly describe the dataset used for the CLIR experiments and the experimental setup.

Language	No. of Documents	No. of Unique Words	Text Size (gzip MB)
English	125516	299689	122
Hindi	95215	208969	110
Marathi	97770	856430	104

Table 4. *Corpora statistics*

7. CLIR Experimental Setup

We conducted CLIR experiments using the corpora released at the FIRE [11] workshop, 2008. The corpora consisted of comparable news articles in English, Hindi and Marathi collected during a span of four years from 2004 to 2007. A sample test topic in English is shown in Figure 5 and the details of the corpora used are mentioned in Table 4.

Each document in the FIRE corpora consists of a unique ID mentioned within the <DOCNO></DOCNO> tags, and the document contents enclosed within the <TEXT></TEXT> tags. The filename of the document is used as the unique ID. This document format is found to be maintained across all language corpora.

Since it was not practically feasible to use all the transliteration systems at runtime or (i.e., online), we prepared a list of extracted named entities from the queries and transliterated them offline. While running the IR experiments, we first checked if a

11. Forum for Information Retrieval Evaluation. http://www.isical.ac.in/člia/

transliteration was available from the intended system. As the queries also had words other than these named entities and also other than the words which were found in the small bilingual dictionaries, we used DATM as the fallback system for all the four methods for such left out words. We chose DATM because we took it as the baseline system and the other three systems were expected to perform better than this.

For the other language pair that we tested on, i.e., English-Marathi, we used exactly the same setup. It may be noted that both Hindi and Marathi use the same script, viz Devanagari.

Next section presents the results of the experiments.

	Method	**MAP**	**R-prec**	**P5**	**P10**	**P15**	**P20**	**P30**
E-H	SMT	0.1523	0.1727	0.2308	0.2214	0.2190	0.2095	0.2080
	SLFSM	0.1868	0.2108	0.3280	0.2940	0.2707	0.2677	0.2532
E-M	SMT	0.1286	0.1345	0.1777	0.1568	0.1498	0.1345	0.1290
	SLFSM	0.1660	0.1825	0.2706	0.2678	0.2508	0.2355	0.2199

MAP: Mean Average Precision
RP: R-prec (Precision over all the retrieved documents)
SLFSM: SMT + LFSM
E: English, H: Hindi, M: Marathi
P1, P5, P10, P15, P20, P30: Precision over the top 1, 5, 10, 15, 20 and 30 documents that are retrieved, respectively

Table 5. *CLIR evaluation using different transliteration systems: the results are significantly better when transliteration as alignment is used for query translation, instead of transliteration as generation, both in terms of the MAP score and R-precision and for both the language pairs.*

8. Evaluation for the CLIR System

The set of 50 test topics in English provided for the FIRE-shared task is used for evaluation. A sample test topic in English is shown in Figure 5. Since our main motivation in this CLIR evaluation is to test the performance of the different transliteration methods, we did not focus on improving the IR results as such. We experimented with two different transliteration methods in the CLIR scenario. The query translation used by us relies mainly on the transliteration module.

In Table 5, we present the CLIR results for one generation based method (SMT) and one alignment based method (SMT+LFSM). As the results show, adding alignment to generation for transliteration can increase the performance of a CLIR system by as much as 22.66% for English-Hindi and by 29.08% for English-Marathi. The increase is significantly more for English-Marathi. This can be explained by the fact that Marathi is morphologically richer than Hindi, which leads to fewer correct translations from the bilingual dictionary and hence the lower results for CLIR. However, when we use alignment rather than generation, the translation is still at the same level,

but the names and foreign words get transliterated better. This explanation is validated by the observation that the absolute results (for both the methods) are lower for English-Marathi, but the increase in performance in the case of alignment is higher.

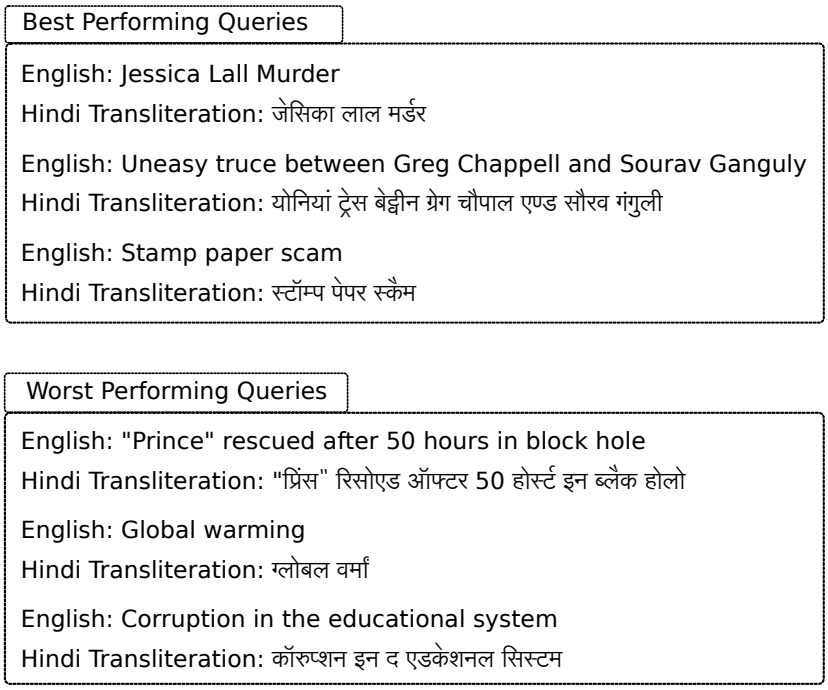

Figure 6. *The best and the worst performing queries and their transliterated versions (prior to being fed to the query translation module). The three major differences among these two kinds of queries are the number of named entities or loanwords, availability of translations in the dictionary and the quality of transliteration. The last one, in the case of alignment, only matters when a query term is not present in the transliterated form in the relevant document.*

As an upper baseline, we used monolingual IR. The MAP scores that we obtained for Hindi-Hindi and Marathi-Marathi were 0.2922 and 0.3055, respectively. The best results for our crosslingual experiments touch 57% of the upper baseline, which is a reasonably good performance.

Figure 4 shows the best and worst performing queries (three each) from the test data for the alignment case. There are three major ways in which the best performing queries differ from the worst ones. The first is that the former contain at least two named entity words (Jessica, Lall, Greg, Chappell, Sourav, Ganguly) or words used, in the given context, as loanwords (stamp, paper, scam) which only need to be transliterated, whereas the latter do not have any such words. The second is that one or more words were found in dictionary in case of the former, but none were found

for the latter. The third is the quality of transliteration. Since a fuzzy string matching based alignment ensures that, if a transliteration is present in the document, it will almost always be matched, the only case in which transliteration is bad enough to affect retrieval is when the word in the query is not present in the transliterated form in the document (and is also not available in the dictionary) and its transliteration aligns with some wrong word, e.g. “warming” aligning to (what can be transliterated in the Latin script as) “verma” (see the second worst performing query in Figure 4). This demonstrates what we discussed earlier, i.e., that correct transliteration of query terms is crucial for CLIR, especially when the resources like the bilingual dictionary are highly inadequate. It also indicates that the need to ensure that wrong alignments have to be minimized. This can partly be achieved by better tuning of the alignment technique.

9. Conclusion

We discussed the importance of transliteration for text processing in general and CLIR in particular, with special focus on the case of English to Indian languages. We argued that the high level of ambiguity (on the source as well as the target side) in this case makes the task of transliteration (and hence CLIR) quite a hard one. We considered six different methods for transliteration and compared the results obtained for ACL NEWS shared task data and presented some observations about combining statistical methods with other kind of methods. Some of the transliteration results are better than (or at least comparable to) the state-of-the-art. We also presented some suggestions, one of them being that we need a better way to evaluate transliteration systems for cases like ours. We then used the output of two transliteration systems (one based on generation and one on generation plus alignment) for English to Indian language CLIR on the FIRE-shared task data and presented the results. Addition of the alignment step in the transliteration system led to a significant increase in the performance of the CLIR system. The increase (over generation alone) was 22.66% for English to Hindi and 29.08% for English to Marathi, although absolute results were lower for English to Marathi as Marathi is a morphologically richer language.

10. References

Abdul Jaleel N., Larkey L. S., “Statistical Transliteration for English-Arabic Cross Language Information Retrieval”, *CIKM ’03: Proceedings of the Twelfth International Conference on Information and Knowledge Management*, ACM, New York, NY, USA, p. 139-146, 2003.

Al-onaizan Y., Knight K., “Machine Transliteration of Names in Arabic Text”, *In ACL Workshop on Comp. Approaches to Semitic Languages*, p. 34-46, 2002.

Aswani N., Gaizauskas R., “A Hybrid Approach to Align Sentences and Words in English-Hindi Parallel Corpora”, *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Association for Computational Linguistics, Ann Arbor, Michigan, p. 57-64, June, 2005.

Cheng P.-J., Teng J.-W., Chen R.-C., Wang J.-H., Lu W.-H., Chien L.-F., "Translating Unknown Queries with Web Corpora for Cross Language Information Retrieval", *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, NY, USA, p. 146-153, 2004.

Davis M. W., Ogden W. C., "Free Resources and Advanced Alignment for Cross-Language Text Retrieval", *Proceedings of TREC*, p. 385-395, 1997.

Ekbal A., Naskar S. K., Bandyopadhyay S., "A Modified Joint Source-Channel Model for Transliteration", *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, Association for Computational Linguistics, Morristown, NJ, USA, p. 191-198, 2006.

Ganesh S., Harsha S., Pingali. P., Varma V., "Statistical Transliteration for Cross Language Information Retrieval using HMM Aligment and CRF", *Proceedings of the 2nd Workshop on Cross Lingual Information Access (CLIA) Addressing the Information Need of Multilingual Societies*, ACL, 2008.

Goto I., Kato N., Uratani N., Ehara T., "Transliteration Considering Context Information Based on the Maximum Entropy Method", *Proceedings of MT-Summit IX*, New Orleans, USA, 2003.

Janarthanam S. C., Subramaniam S., Nallasamy U., "Named Entity Transliteration for Cross Language Information Retrieval Using Compressed Word Format Mapping Algorithm", *iNEWS '08: Proceeding of the 2nd ACM Workshop on Improving Non-English Web Searching*, ACM, New York, NY, USA, p. 33-38, 2008.

Knight K., Graehl J., "Machine Transliteration", *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, p. 128-135, 1997.

Li H., Zhang M., Su J., "A Joint Source-Channel Model for Machine Transliteration", *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, p. 159, 2004.

Llitjos A., Black A., "Knowledge of Language Origin Improves Pronunciation of Proper Names", *Proceedings of EuroSpeech*, vol. 1, p. 1919-1922, 2001.

Malik M. G. A., "Punjabi Machine Transliteration", *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, p. 1137-1144, 2006.

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

Och F. J., Ney H., "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics*, vol. 29, n° 1, p. 19-51, 2003.

Oh J., Choi K., "An English-Korean Transliteration Model Using Pronunciation and Contextual Rules", *Proceedings of the 19th International Conference on Computational Linguistics*, vol. 1, p. 1-7, 2002.

Qu Y., Grefenstette G., "Finding Ideographic Representations of Japanese Names Written in Latin Script via Language Identification and Corpus Validation", *Proceedings of ACL*, p. 183-190, 2004.

Qu Y., Grefenstette G., Evans D. A., "Automatic Transliteration for Japanese-to-English Text Retrieval", *Proceedings of SIGIR*, p. 353-360, 2003.

Sato S., "Crawling English-Japanese Person-Name Transliterations from the Web", *WWW '09: Proceedings of the 18th International Conference on World Wide Web*, ACM, New York, NY, USA, p. 1151-1152, 2009.

Singh A. K., "A Computational Phonetic Model for Indian Language Scripts", *Proceedings of Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands, October, 2006.

Singh A. K., "Study of Some Distance Measures for Language and Encoding Identification", *Proceedings of ACL 2006 Workshop on Linguistic Distance*, Sydney, Australia, 2006a.

Singh A. K., Surana H., Gali K., "More Accurate Fuzzy Text Search for Languages Using Abugida Scripts", *Proceedings of ACM SIGIR Workshop on Improving Web Retrieval for Non-English Queries*, Amsterdam, The Netherlands, 2007.

Subramaniam S., Singh A. K., Dasigi P., "Experiments in CLIR Using Fuzzy String Search Based on Surface Similarity", *Proceedings of the 32nd Annual ACM SIGIR Conference*, Boston, Massachusetts, 2009.

Surana H., Singh A. K., "A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages", *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, Asian Federation of Natural Language Processing, Hyderabad, India, 2008.

Virga P., Khudanpur S., "Transliteration of Proper Names in Cross-Lingual Information Retrieval", *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition*, Association for Computational Linguistics, Morristown, NJ, USA, p. 57-64, 2003.