Automated Mining Of Names Using Parallel Hindi-English Corpus

R. Mahesh K. Sinha
Indian Institute of Technology, Kanpur, India
rmk@iitk.ac.in

Abstract

Machine transliteration has a number of applications in a variety of natural language processing related tasks such as machine translation, information retrieval and question-answering. For automated learning of machine transliteration, a large parallel corpus of names in two scripts is required. In this paper we present a simple yet powerful method for automatic mining of Hindi-English names from a parallel corpus. An average 93% precision and 85% recall is achieved in mining of proper names. The method works even with a small corpus. We compare our results with Giza++ word alignment tool that yields 30% precision and 63% recall on the same corpora. We also demonstrate that this very method of name mining works for other Indian languages as well.

1 Introduction

Transliteration of names from one script/language to another has a number of applications in a variety of natural language processing tasks. These include machine translation, information retrieval, question-answering, multilingual directories, reservation charts, name lists etc.

Machine transliteration has been studied by a number of researchers (Knight et al., 1998; Al-Onaizan et al., 2002; Goto et al., 2003; Huang et al., 2003; Feng et al., 2004; Asif et al., 2006; Kuo et al. 2006); Knight and Graehl(1998) use a modular approach in which five probability distributions are obtained for various phases of the transliteration - generation and pronunciation of English word sequences, conversion of English sounds to Japanese and then Japanese sounds to Katakana writing. Al-Onaizan and Knight (2002) present work on transliteration from English to Arabic. It relies on an existing named entity recognition system, which identifies possible named entities in English. A predefined phoneme mapping is used to generate all possible transliterations. The validity of transliterations is examined by rating it based on web counts, and co-references by querying for the candidate transliteration on popular search engines such as Google. Huang et al. (2003) have worked on extracting Hindi-English named entity pairs through alignment of a parallel corpus. Chinese-English pairs are first extracted using a dynamic programming string matching. This Chinese-English model is then adapted to Hindi-English iteratively, by using already extracted Hindi-English named entity pairs to bootstrap the model. The precision achieved by this model is 91.8%. Feng et al. (2004) have used a maximum entropy model, in which an alignment probability for target/source named entities is defined over 4 features - translation score, transliteration score, co-occurrence score and distortion score. The extraction of each feature is involved, but the maximum entropy model over these features is straightforward. Kuo et al. (2006) uses a syllable alignment algorithm for cross-language syllable-phoneme conversion. Asif et al. (2006) have considered Bengali to English transliteration. They present a model which upon supervised training provides direct orthographical mapping. They report an accuracy of 69-89%. The success of all of these works depends upon the volume and nature of name corpora used.

In this paper, we present a simple yet powerful method for mining of Hindi-English names from a parallel text corpus. In Hindi, the words are written as they are spoken i.e. it is phonetic in nature. On the other hand, English is non-phonetic in the sense that there is a specified usage of a spelling for every word. Hindi names when written in English have a similar problem that the users have developed their own spellings for names that are commonly accepted. Though these English spellings do retain the phonetic structure of Hindi to a large extent, there are variations that cannot be easily captured through rules. In table 1 a few illustrative examples are given. It is evident that the Hindi vowel modifiers (called 'matra') do not have unique mappings to English vowel combinations. It is difficult to derive simple mapping rules for these. The map-

Proceedings of the 7th Workshop on Asian Language Resources, ACL-IJCNLP 2009, pages 48–54,
Suntec, Singapore, 6-7 August 2009.

ping of semivowels ‘y’ and ‘v’ and ‘schwa’ deletions are highly contextual. However, for the consonants, the mappings are straightforward barring a few exceptions.

Our strategy for automatic mining of Hindi-English proper names from parallel corpus exploits this near-invariance in consonant mapping. We compare our results with Giza++ word alignment. In the following section, we present our design methodology followed by experimental results and conclusions.

Hindi word in Devanagari	Hindi word in IITK-Roman (Appendix-A)	Corresponding commonly used English (Roman) transliteration	Unacceptable English (Roman) transliterations	Observations
हरीश	harISa	Harish	Hareesh / Hariesh / Hareish	i. long vowel mapping ii. ‘schwa’ deletion iii. consonant cluster mapping
संजीव	saMjIva	Sanjeev or Sanjiv	Sanjiiv / Sanjiev /Sanjeiv	i. variation in long vowel mapping ii. ‘schwa’ deletion
फाल्गुनी	PAlgunI	Phalguni	Falguni	i. long vowel mapping ii. consonant mapping
मूना	mUnA	Moona	Muna / Muuna / Moonaa	preferred long vowel mapping
सूरज	sUraja	Suraj	Sooraj / Suuraj / Suraz /Surag	i. long vowel mapping ii. ‘schwa’ deletion iii. consonant mapping
सोमनाथ	somanAWa	Somenath or Somnath	Somanath / Somanaath	i. long vowel mapping ii. ‘schwa’ deletion iii. peculiar vowel mapping to ‘e’
सक्सेना	saksenA	Saxena	Saksena	i. long vowel mapping ii. preferred consonant mapping
दीक्षित	xIkSiwa	Dixit or Dikshit	Deexit / Dikchhit etc.	i. long vowel mapping ii. ‘schwa’ deletion iii. preferred consonant mapping
मोदी	moxI	Modi	Modee / Modii / Mody etc.	preferred long vowel mapping
सोनिया	soniyA	Sonia	Soniya	preferred semivowel mapping
रामदेव देव	rAmaxeva xeva	Ramdeo Deva	Ramdev /Ramadev / Ramadeo Deo / Dev	preferred semivowel mapping

Table 1: An Illustration of Hindi to English Name Transliteration Variations

2 Hindi-English Name Corpus Creation

We use an aligned parallel Hindi-English text corpus for creation of Hindi-English name corpus. The size of the corpus is immaterial and it could be as small as a few lines. The sentence alignment also need not be perfect as long as the aligned set of sentences contain the translated sentences. Our methodology is even capable of capturing to some extent mapping between old city names with new city names such as Bombay and Mumbai. Figure 1 depicts the process of name mining diagrammatically.

The Hindi text written in Devanagari is first converted to IITK-Roman form (appendix-A). IITK-Roman has become a de-facto standard used by a large number of researchers in India. The conversion to IITK-Roman form is straightforward and is a direct representation of UTF-8 or ISSCII-8 coding schemes without any

loss of constituent information in terms of phonemes or constituent symbols. The usage of IITK-Roman form is more for entry and programming convenience.

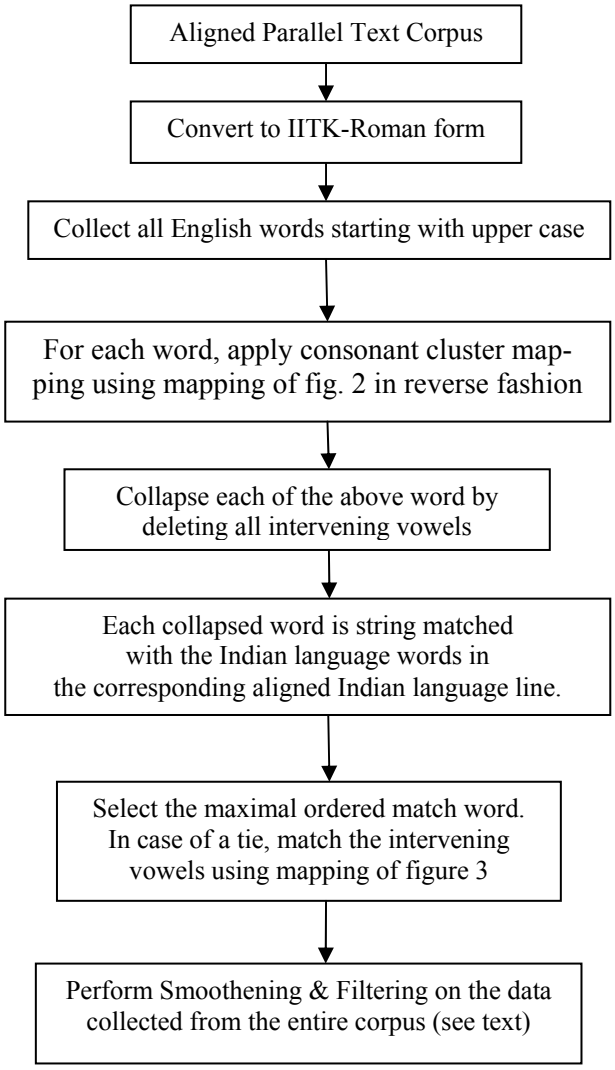

Figure 1: Schematic flow diagram of the name mining process

As outlined earlier, in order to simplify the learning process, the trivial consonant (C) and consonant cluster (C^{+}) mappings are provided separately in the form of rules. The main consonant mappings from IITK-Roman to English are shown in figure 2.

k(क)→k/c/ck; K(ख)→kh; g(ग)→g; G(घ)→gh; f(ङ)→n;

c(च)→ch; C(छ)→chh; j(ज)→j/z; J(झ)→jh; F(ञ)→n;

t(ट)→t; T(ठ)→th; d(ड)→d; D(ढ)→dh; N(ण)→n;

w(त)→t; W(थ)→th; x(द)→d; X(ध)→dh; n(न)→n;

p(प)→p; P(फ)→ph/f; b(ब)→b; B(भ)→bh; m(म)→m;

y(य)→y; r(र)→r; l(ल)→l; v(व)→v/w;

s(स)→s; S(श)→sh; R(ष)→sh; h(ह)→h;

kR(क्ष)→x; jF(ज्ञ)→ gy; dZ(ड़) →r;

q (ृ)→r/k; M(ं)→n; H(ः)→h;

ks(क्स)→x; kZ (क़)→q; jZ (ज़)→z; PZ (फ़)→f

Figure 2: IITK-Roman to English consonant mapping

A (ा)→ a; i (ि)→ i; I (ी)→ i; u (ु)→u;

U(ू)→u; e(े)→e; E(ै)→ai; o (ो)→o;

O(ौ)→ou;

Figure 3: IITK-Roman to English vowel mapping

The consonant mappings are exploited in hypothesizing plausible name transliterations. Following steps explain the process of mining of Hindi-English name pairs:

i. For each aligned line, collect all the words in the English sentence that have first letter in upper case. These are potential English proper names excepting the first word that may or may not be a proper name.

ii. For each word, apply consonant cluster map-ping from English to Hindi (using the mapping as given in figure 2 in reverse fashion). In absence of a defined mapping, the consonant is ignored. This yields one or more plausible Hindi names as there are one to many reverse mappings. The following three mappings are very rare and so are ignored for efficiency: f→n; F→n; H→h. Further, the semivowel 'y' is not treated as a consonant if it is the last character of the word. It is treated as a consonant if it is preceded or followed by a vowel.

iii. Collapse each of the above word into being part of the plausible Hindi name by deleting all vowels in it.

iv. Each collapsed plausible Hindi name, as de-rived in the preceding step, is string-matched with the Hindi words in the corresponding aligned Hindi line. The process of matching looks for maximal ordered string match omitting the Hindi vowels.

- In case no match is found, it is ignored.
- In case of multiple matches, minimal word length distance is taken as the criterion for selection.

- In order to avoid false matching, length must be greater than 1 and at least 30% of characters must match.
- Further, a constraint that the first character of the mapped words must both be either a consonant or both be a vowel, is imposed.

v. In case two or more matches have same maximal length match, then the maximal match with the plausible un-collapsed (i.e. including the intervening vowels with their mapping using figure 3) Hindi name is matched and the ordered maximal length match is selected. Usually such a situation is encountered when two or more similar names are encountered in the aligned lines. An example of this would be say the two names 'Hindi' and 'Hindu' occur in the same sentence. These will get matched to the same degree by step (iv) above. The way to resolve this is to also take intervening vowels into account. The IITK Roman vowel mapping to English used here is given in figure 3. It may be noted that only one vowel mapping out of the many possibilities, has been taken. This is the most frequent mapping and is taken as the baseline vowel mapping.

vi. The final stage is that of filtering and smoothening.

- For every English name, the corresponding Hindi name mapping(s) with their frequency of occurrence is recorded for the entire corpus.
- In case of multiple mappings, each mapping is examined. The suffix that represent the post-position markers such as ne (ne ने), ka(kA का), ko (ko को), ki(kI की), ke(ke के), se(se से), men(meM में), par(para पर), vala (vAlA वाला) etc. in Hindi are stemmed. Further, other morphological co-joiners ('sandhi') for other Indian scripts are also stemmed.
- After stemming, the frequency is recomputed.
- The mapping with the highest frequency is selected.

Although these post-position markers in Hindi are separate words and are usually written with a preceding blank, many a time it is not properly observed and appears as a suffix.

Given below is an illustrative example:
English sentence:
It goes daily from Delhi to Mumbai, Bangalore, Varanasi and Lucknow.
Aligned Hindi Sentence:
यह रोजाना दिल्ली से मुम्बई, बैंगलुरु, वाराणसी और लखनऊ जाती है ।
(Converted to IITK-Roman)
yaha rojAnA xillI se mumbaI, bEMgaluru, vArANasI Ora laKanaU jAwI hE.
Probable English Proper Nouns:
It Delhi Mumbai Bangalore Varanasi Lucknow
Plausible Hindi Names after reverse consonant substitutions:
{it iw} {delhi xelhi} {mumbai} {bangalore baMgalore} {varanasi varaNasi varaMasi}{luknov lukNov lukMov}
Collapsed plausible corresponding Hindi Names:
{t w} {dlh xlh} {mmb} {bnglr bMglr} {vrns vrNs vrMs} {lknv lkNv lkMv}
Hypothesized Hindi Names after matching:
Delhi→ xillI दिल्ली *;*
Mumbai →mumbaI मुम्बई*;*
Bangalore →bEMgaluru बैंगलुरु*;*
Varanasi → vArANasI वाराणसी*;*
Lucknow →laKanaU लखनऊ.
In the above example, the first word 'It' does not get matched to any of the Hindi words because of the constraint that the matching length has to be greater than 1 and a minimum of 30% of length must match.

It is interesting to note the method outlined captures even those names that differ in their forms or spelling such as Delhi & दिल्ली (xillI), Bangalore & बैंगलुरु (bEMgaluru) and Lucknow & लखनऊ (laKanaU) based on maximal match. For transliteration, these have to made table driven.

Given below is an illustration of step (v) of the procedure:
English sentence:
Mr. Handa speaks Hindi and he is a Hindu.
Aligned Hindi Sentence:
श्री हांडा हिन्दी बोलते हैं और वह एक हिन्दू हैं ।
(Converted to IITK-Roman)
SrI hAMdA hinxI bolawe hEM Ora vaha eka hinxU hEM.

Probable English Proper Nouns:

Mr Handa Hindi Hindu.

Plausible Hindi Names after reverse consonant substitutions:

{mr mq} {haNda handa haMda haNxa hanxa haMxa} {hiNdi hindi hiMdi hiNxi hinxi hiMxi} {hiNdu hindu hiMdu hiNxu hinxu hiMxu}

Collapsed plausible corresponding Hindi Names:

{mr mq} {hNd hnd hMd hNx hnx hMx} {hNd hnd hMd hNx hnx hMx} {hNd hnd hMd hNx hnx hMx}

Hypothesized Hindi Names after matching:

Handa→ hAMdA हांडा*; hinxI* हिन्दी; *hinxU* हिन्दू;

Hindi → hAMdA हांडा*; hinxI* हिन्दी; *hinxU* हिन्दू;

Hindu → hAMdA हांडा*; hinxI* हिन्दी; *hinxU* हिन्दू;

Now since these are equiprobable multiple matches, step (v) will get invoked. For each matching target word, the vowel mapping of figure 3 is applied. This yields the following:

hAMdA हांडा*→ haMda;*

hinxI हिन्दी*→hinxi*;

hinxU हिन्दू*→hinxu*;

Now the English source word is matched and minimal distance word is selected. This finally yields the desired result as follows:

Handa→ hAMdA हांडा*;*

Hindi → hinxI हिन्दी;

Hindu → hinxU हिन्दू;

Given below is an illustration of step (vi) of the procedure:

Suppose in the entire corpus the city name 'Agra' yields the following matches:

i. Agra →*AgarA* आगरा; count=20;

ii. Agra →*Agare* आगरे; count=12;

iii. Agra →*AgarAse* आगरासे; count=5;

iv. Agra →*AgarAmeM* आगरामें; count=4;

v. Agra →*AgarAkA* आगराका; count=2;

Now the process of smoothening will convert *AgarAse* आगरासे to *AgarA* आगरा by deleting post-position suffix 'se'से; *AgarAmeM* आगरामें to *AgarA* आगरा by deleting post-position suffix 'meM'में; and *AgarAkA* आगराका to *AgarA* आगरा by deleting post-position suffix 'kA'का. This will yield the final table as follows:

i. Agra →*AgarA* आगरा; count=31;

ii. Agra →*Agare* आगरे; count=12;

The filtering process will select the mapping of Agra →*AgarA* आगरा.

It may be noted that the word *Agare* आगरे is the oblique form of the name *AgarA* आगरा and such usage is very common in Indian languages. A morphological processing is required to make the conversion and this has not been implemented in the current implementation.

3 Experimentation and Results

For experimentation, we took a text that contained a lot of names. Two sentence aligned files were created from a Indian freedom fighters' story. This story contains a lot of names of individuals and places in the text. The results of our name mining methodology are summarized in table 2. We also used Giza++ word alignment tool (Och and Ney, 2003) on the same files and collected figures pertaining to the alignment of proper names in Hindi and English. In case of multiple mappings for a proper name in which one of them is a correct mapping, it is considered as 'false positive'. These results are also shown in table 2 for comparison.

	File1		File2	
	Name-mapping	Giza++	Name-mapping	Giza++
Total no. of words	2439	2439	4909	4909
Total no. of Names(N)	192	192	343	343
Correct mapping (TP)	155	57	262	74
Incorrect mapping (FP)	13	117	35	200
Not-captured (FN)	24	18	46	69
Accuracy (TP/N)	0.8073	0.2969	0.7638	0.2157
Precision (TP/(TP+FP))	0.9226	0.3276	0.9495	0.2701
Recall (TP/(TP+FN))	0.8659	0.7600	0.8506	0.5175
F-measure (2PR/(P+R))	0.8934	0.4578	0.8968	0.3549

Table 2. Result for name mining and word-alignment algorithms.

Our experimentation reveals that our name mining methodology yields a precision of 92 to 95% and a recall of 85 to 86% resulting in F-measure of 0.89. On the other hand, the Giza++ word alignment tool yields a precision of 27 to 33% and a recall of 52 to 76% resulting in F-measure of 0.35 to 0.46. The results are a clear demonstration of effectiveness our approach of mining proper names from the parallel Hindi-English corpora. Most of the errors using our approach have been found to be due to short names, words not properly delineated in the target text, morphological changes in the target text, the first word in English not being a proper noun or different forms of names that are used denoting the same place. It should be noted that our approach works even for a corpus of a few lines as it is primarily a rule-based method.

The method as outlined above is equally applicable to other Indian languages. In order to demonstrate this, we conducted a limited experiment with Punjabi and Bengali languages. A corpus of about 200 sentences was taken. The same program as was used for Hindi with no change in the mapping tables was used for the experimentation. The results obtained were remarkable and a performance of about 90% and 70% of correct mining of proper names for Punjabi and Bengali respectively is yielded. The poorer performance in case of Bengali is primarily due to morphological changes that take place in the proper names based on their role in the sentence. Unlike in Hindi where the post-positions are written separately or simply suffixed, for most of the other Indian languages, these post-position markers are co-joined ('Sandhi') with the preceding word leading to a morphological change. This is less frequent in Punjabi. Further, Bengali has no consonant for 'va' ব and this is mapped to 'ba' ব. So some consonant mapping changes are required to yield better results for another Indian language but the methodology remains the same. Here are some example mappings:

Bengali:

i. Cath hasn't phoned since she went to Berlin.
bArline yAoyZA Weke kyAWa Pona kareni।
বার্লিনে যাওয়া থেকে ক্যাথ ফোন করেনি।

ii. Jo was the next oldest after Martin.
mArtinera parei badZa Cila jo।
মার্টিনের পরেই বড় ছিল জো।

Names extracted:
Cath → kyAWa ক্যাথ;
Berlin → bArline বার্লিনে
Here the correct mapping is 'bArlina বার্লিন' but the name has got morphologically transformed to 'bArline বার্লিনে' (to Berlin) based on co-joining of post-position marker.
Martin → mArtinera মার্টিনের
Here the correct mapping is 'mArtina মার্টিন' but the name has got morphologically transformed to 'mArtinera মার্টিনের' (after Martin)) based on co-joining of post-position marker.

Punjabi:
i. Sam Sand Dunes is one of the best nature's gift to the human beings.
sEma sEzda diUnasa manuYKa xe laI prakirawI xe saraba SreSata wohaPZiAz viYcoz iYka hE.
ਸੈਮ ਸੈਂਡ ਡਿਊਨਸ ਮਨੁੱਖ ਦੇ ਲਈ ਪ੍ਰਕਿਰਤੀ ਦੇ ਸਰਬ ਸ਼੍ਰੇਸ਼ਟ ਤੋਹਫ਼ਿਆਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ।

ii. Bikaner is located to the north of Rajasthan popularly known as a camel country.
bIkAnera rAjasaWAna xe uYwara viYca saWiwa hE awe saXAraNa wOra we UTa-praxeSa xe rUpa viYca jANiA jAzxA hE.
ਬੀਕਾਨੇਰ ਰਾਜਸਥਾਨ ਦੇ ਉੱਤਰ ਵਿੱਚ ਸਥਿਤ ਹੈ ਅਤੇ ਸਧਾਰਣ ਤੌਰ ਤੇ ਊਠ-ਪ੍ਰਦੇਸ਼ ਦੇ ਰੂਪ ਵਿੱਚ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

Names extracted:
Sam → sEma ਸੈਮ ;
Sand → sEzda ਸੈਂਡ ;
Dunes → diUnasa ਡਿਊਨਸ ;
Bikaner → bIkAnera ਬੀਕਾਨੇਰ ;
Rajasthan → rAjasaWAna ਰਾਜਸਥਾਨ

4 Conclusions

In this paper, we have presented a simple yet powerful method for mining of Hindi-English proper name corpus with a success of mining being 93% precision. In contrast, GIZA+ word alignment tool on same sized corpus yielded 29% precision. The proposed method works even for a single line text. Moreover, there is no strict requirement of sentence alignment as it works equally well for one to many and many to many sentence alignment as long as the target group of sentences contain the corresponding translation.

Thus it works under noisy environments where sentence boundaries are not correctly identified. Our approach also yields a table of similar old city names with new city names that is very frequently encountered in Indian context.

The methodology outlined in this paper for automatic mining of proper names are equally applicable to all Indian languages as all Indian scripts are phonetic in nature in the same way as Devanagari (used for Hindi). We have also demonstrated that this very method of name mining without making any changes in the program or the mapping table as used for Hindi, works for other Indian languages. Our limited experimentation for Punjabi and Bengali and have yielded performance of 90% and 70% respectively of correct mining of proper names.

There are several other advantages of our approach. Since the proper name mining is captured with a high accuracy over a rough or noisy aligned corpus, it is possible to use these as anchors (the same way as numerals) for improvement of the alignment results. These anchors will also be useful in word alignment programs for speedy convergence. Accurate word alignment is crucial to the success of any statistical machine translation system. Another byproduct of our approach is that it also yields the table of old city names with new city names. In India, a large number of city names that were used during British time, have undergone a change and most of these changes are phonetic variations of the old names.

Acknowledgements

Author is thankful to Saleem Siddiqui and Abhay Singh for experimentation and testing.

References

Al-Onaizan Y. and Knight K.2002. Translating Named Entities Using Monolingual and Bilingual Resources. *Proceedings of ACL 2002*, 400-408.

Ekbal Asif, Sudip Kumar Naskar and Sivaji Bandyopadhyay. 2006. A Modified Joint Source-Channel Model for Transliteration, *Proceedings of ACL 2006.*

Feng Dong-Hui, Ya-Juan Lv, and Ming Zhou. 2004.A New Approach for English-Chinese Named Entity Alignment. *Proceedings of ACL 2004.*

Goto I., N. Kato, N. Uratani, and T. Ehara. 2003. Transliteration considering Context Information based on the Maximum Entropy Method. *Proceeding of the MT-Summit IX*, New Orleans, USA, 125-132.

Huang Fei, Stephan Vogel, and Alex Waibel. 2003. Extracting Named Entity Translingual Equivalence with Limited Resources. *ACM Transactions on Asian Language Information Processing (TALIP),* 2(2):124–129.

Knight K. and J. Graehl. 1998. Machine Transliteration, *Computational Linguistics*, 24(4): 599-612.

Kuo Jin-Shea , Haizhou Li and Ying-Kuei Yang. 2006. Learning Transliteration Lexicons from the Web, *The 44th Annual Meeting of Association for Computational Linguistics (COLING-ACL2006)*, Sydney, Australia, 1129 – 1136.

Och Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models, *Computational Linguistics*, 29(1):19-51. (http://www.fjoch.com/GIZA++.html)

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bar. 1994. Algorithms for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2): 183-193.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of Proper Names in Crosslingual Information Retrieval. *Proceedings of the ACL 2003 Workshop on Multilingual and Mixedlanguage Named Entity Recognition*, Sapporo, Japan, 57-60.

Appendix-A: IITK-Roman code for Hindi (Devanagari)

अ आ इ ई उ ऊ ऋ ए ऐ ओ औ

ा ि ी ु ू ृ े ै ो ौ ं ः ँ ॅ ॉ

a A i I u U q e E o O M H V z Z

क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ द ध न

k K g G f c C j J F t T d D N w W x X n

प फ ब भ म य र ल व स श ष ह

p P b B m y r l v s S R h