Hindi to English Machine Transliteration of Named Entities using Conditional Random Fields

Manikrao L Dhore
Vishwakarma Institute of Technology, Pune, India.

Shantanu K Dixit
Walchand Institute of Technology, Solapur, India.

Tushar D Sonwalkar
Vishwakarma Institute of Technology, Pune, India.

ABSTRACT
Machine transliteration has received significant research attention in recent years. In most cases, the source language has been English and the target language is an Asian language. This paper focuses on Hindi to English machine transliteration of Indian named entities such as proper nouns, place names and organization names using conditional random fields (CRF). Hindi is the national language of the India and spoken by more than 500 millions Indian. Hindi is the world's fourth most commonly used language after Chinese, English and Spanish. This system takes Indian place name as an input in Hindi language using Devanagari script and transliterates it into English. The input to the system is provided in the form of syllabification in order to apply the n-gram techniques. As more than 50% named entities are formed as a combination of two and three syllabic units, the n-gram approach with unigrams, bigrams and trigrams of Hindi are used to train the corpus. The system provides the satisfactory performance for trigrams as compared to unigrams and bigrams.

General Terms
Machine Transliteration

Keywords
Bigram, Conditional Random Fields, Trigram, Transliteration, Syllabification

1. INTRODUCTION
It is challenging to translate names and technical terms occurring in the user input across languages with different alphabets and sound inventories. One of the most frequent problems translators must deal with is translating proper names and technical terms in the user input. These items are commonly transliterated, i.e., replaced with approximate phonetic equivalents. For example, a place name "कंचनपुर" in Devanagari language is transliterated as "Kanchanpur" in English. Translating such items from English back to Devanagari language is even more challenging, and of practical interest, as transliterated items make up the bulk of text phrases not found in bilingual dictionaries.

The major challenge is the transliteration of out of vocabulary (OOV) words appearing in the user input. The major portion of user input consists of named entities, numbers, acronyms and technical terms. These words can be the most important words in the user input. These words need to be transcribed into the document language when the query and document languages do not share a common alphabet. The practice of transcribing a word or text written in one language into another language is called *transliteration*. Transliteration is the conversion of a word from one language to another without losing its phonological characteristics [1]. Phonetic translation across these pairs is called transliteration. Transliteration is the conversion of a given name in the source language (a text string in the source writing system or orthography) to a name in the target language (another text string in the target writing system or orthography), such that the target language name is:

- phonemically equivalent to the source name
- conforms to the phonology of the target language and
- matches the user intuition of the equivalent of the source language name in the target language [2]. It is the practice of transcribing a word or text written in one writing system into another writing system. Machine transliteration is usually used to support the machine translation (MT) and cross-language information retrieval (CLIR) to convert the named entities.

The direct transliteration of Hindi to English is quite difficult due to the following factors:

- Hindi uses the Devanagari script whereas English uses the Roman script
- The Hindi alphabet contains 52 characters whereas the English alphabet contains only 26 characters.
- There is no concept of capitalization of leading characters of names in Indian languages unlike English and other European languages which plays an important role in identifying named entities (NEs).
- Hindi has highly phonetic characteristics whereas English is not a phonetic language.
- In English proper names are not used as person names whereas in Hindi most of the person names are being used as common names.
- In India place names are frequently homographic with common words or with person names, presence of a number of exonyms (foreign language equivalences), endonyms (local variants) and historical variants for many place names.
- A source language word can have more than one valid transliteration in target language. For example, for the Hindi named entity below four different transliterations are possible:

  लक्ष्मणराव – Lakshmanrāv, Laxmanrāv, Lakshmanrāo and Laxmanrāo

- Unavailability of resources such as the parts of speech tagger (POS) and a good morphological analyzer for Indian Languages (ILs). Name lists are found in webs which are in English but no such lists for Indian Languages can be found in Unicode.
- Hindi language is highly inflectional [3].

From a linguistic point of view there are many issues such as orthographic variations, ambiguous spelling, lexical ambiguity, morphological variations, affixation, root and pattern, tokenization, translation divergences conflation,

source meaning–target meaning, source syntax–target syntax and source word – target word.

2. RELATED WORK

Most of the machine transliteration work outside India is carried out for English to Japanese, English to Chinese, English to Korean, English to Russian, English to Japanese (Katakana), English to Korean Hangul, English to Pinyin, Pinyin to Chinese, Thai to English, Chinese to English, English to Arabic, Arabic to English, English to Thai, Urdu to English, Persian to English, Spanish to Chinese, Japanese to English, Swedish to Finnish, English to Hebrew, English to Spanish and Spanish to English language pairs. For Indian languages English to Hindi, English to Tamil, Shahmukhi to Gurmukhi, English to Telugu, Bengali to English, English to Kannada, English to Oriya, Hindi to English, Punjabi to Hindi language pairs are used.

The grapheme-based and phoneme-based models are used for the machine transliteration. The grapheme based model treats transliteration as an orthographic process and tries to map the source language graphemes directly to the target language graphemes. Conceptually, it is a direct orthographical mapping from source graphemes to target graphemes [4]. Phoneme-based model considers transliteration as a phonetic process. One of the early works on transliteration is done by Arababi in 1994 by combining neural net and expert systems [5]. Knight and Graehl developed a five stage statistical model to do back transliteration, that is, recover the original English name from its transliteration into Japanese Katakana in 1997[6]. Stalls and Knight adapted this approach for back transliteration from Arabic to English of English names in 1998[7]. Al-Onaizan and Knight have produced a simpler Arabic-English transliterator and evaluated how well their system can match a source spelling in 2002. Their work includes an evaluation of the transliterations in terms of their reasonableness according to human judges [8]. Work in the field of Indian Language CLIR was done by Jaleel and Larkey in 2003 which was based on their work in English-Arabic transliteration for CLIR [9]. Their approach was based on Hidden Markov Model using GIZA++. Phoneme-based models, based on weighted finite state transducers [10] and Markov window [11] considers transliteration as a phonetic process. In 2003 the team of National Centre for Software Technology, Mumbai has given a unified table driven approach for conversion between phonemic code and Unicode [12]. OM transliteration scheme provides a script representation which is common for all Indian languages [13]. In 2006, Punjabi machine transliteration for Punjabi language from Shahmukhi to Gurmukhi used the set of transliteration rules [14]. Sproat [15-17] presented a formal computational analysis of Brahmi scripts**.** Kopytonenko [18] also focused on computational models that perform grapheme-to-phoneme conversion. Ganesh, Harsha, Pingali and Verma have developed a statistical transliteration technique that is language independent. They selected a statistical model for transliteration which is based on Hidden Markov Model (HMM) alignment and Conditional Random Fields (CRF) [19]. Sujan Kumar Saha, Partha Sarathi Ghosh, Sudeshna Sarkar, and Pabitra Mitra [20] have proposed a two-phase transliteration methodology in 2008. The transliteration module uses an intermediate alphabet, which is designed by preserving the phonetic properties. Ekbal, Naskar and Bandyopadhyay (2007-2010) made significant attempt to develop transliteration systems for Indian languages to English and especially for Bengali-English transliteration [21-28]. Manoj K. Chinnakotla, Om P. Damani, and Avijit Satoskar have developed a reasonable transliteration system for resource scared languages by judiciously applying statistical techniques to monolingual resources in conjunction with manually created bilingual rule bases in 2010. The statistical technique is the Character Sequence Modeling (CSM), called Language Modeling. They have proved that if the word origin is used for the transliteration, then the system performs better than statistical methods [29]. Jong-Hoon Oh approach is based on two transliteration models. They used three different machine learning algorithms CRF, MIRA and MEM for building multiple machine transliteration engines [30]. Our work is related to the machine transliteration of Hindi to English using CRF. A Hindi-English language is less studied and can be better investigated using CRF. For doing this, we have used n-gram as a feature over the syllabified Hindi input to obtain the transliteration in English.

3. CRF

A CRF is a form of undirected graphical model that defines a single log-linear distribution over label sequences given a particular observation sequence. CRF model defines a conditional probability P(Y|X) over label sequences given a particular observation sequence X, rather than a joint distribution over both label and observation sequences. Formally, we define G = (V, E) to be an undirected graph such that there is a node v ϵ V corresponding to each of the random variables representing an element Y_v of Y. If each random variable Y_v obeys the Markov property with respect to G, then (Y, X) is a conditional random field. In machine transliteration CRF can be used to generate the target language word from a source language word. CRF is defined as conditional probability distributions P(Y |X) of target language words given source language words. The probability of a particular target language word Y given source language word X is the normalized product of potential functions each of the form

$$p_\theta(Y|X) = \exp\Big(\sum_j \lambda_j t_j(Y_{i-1}, Y_i, X, i) + \Big(\sum_k \mu_k s_k(Y_i, X, i)\Big)\Big) \quad (1)$$

where $t_j(Y_{i-1}, Y_i, X, i)$ is a transition feature function of the entire source language word and the target language characters at positions i and i−1 in the target language word; $s_k(Y_i, X, i)$ is a state feature function of the target language word at position i and the source language word; and λ_j and μ_k are parameters to be estimated from training data. The notations can be simplified by writing

$$s_k(Y_i, X, i) = s(Y_{i-1}, Y_i, X, i)$$

and

$$F_j(Y,X) = \sum_{i=1}^{n} f_j(Y_{i-1}, Y_i, X, i) \quad (2)$$

where each $f_j(Y_{i-1}, Y_i, X, i)$ is either a state function $s(Y_{i-1}, Y_i, X, i)$ or a transition function $t(Y_{i-1}, Y_i, X, i)$. This allows the probability of a target language word Y given a source language word X to be written as

$$P(Y|X,\lambda) = \frac{1}{Z(X)} \exp\left(\sum_j \lambda_j F_j(Y,X)\right) \qquad (3)$$

Z(X) is a normalization factor. When applying CRFs to transliteration problem, an observation sequence X is a string of source language transliteration units and state sequence or tag sequence Y is the string of target language transliteration units [31-33].

4. TRANSLITERATION SYSTEM

The overall logical flow of Hindi to English machine transliteration system is depicted in figure 1 and 2.

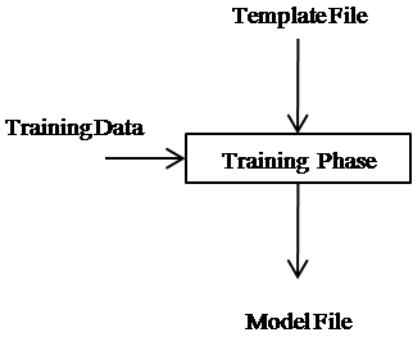

Figure 1. Training of Data Set

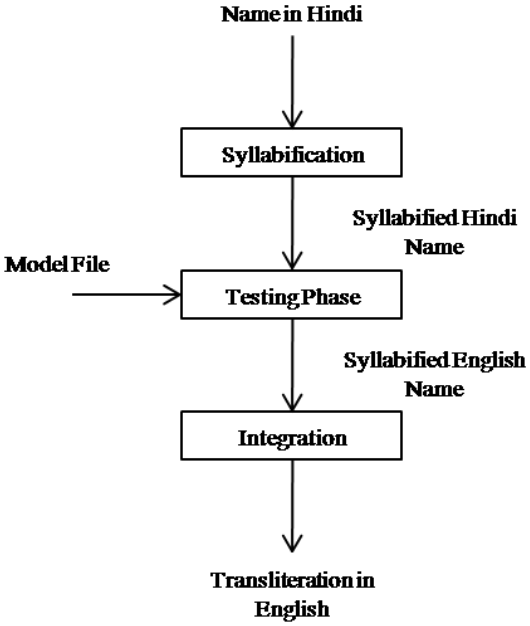

Figure 2. Transliteration System

4.1 ANALYSIS OF NAMED ENTITY FORMATION IN HINDI

Hindi is written using Devanagari script, and the minimum syllabic unit of Devanagari is called as akshara. For example, the named entity "सचिन" – स + चि + न (Sachin) has three aksharas स, चि and न respectively. It has been observed that in Hindi language the minimum length of the named entity is 1 akshara (formed using 1 syllabic unit) and maximum length is 8 aksharas. There are very few named entities consisting one syllable. From the number of aksharas in the named entities, 8 categories are made. One akshara is considered equivalent to one phonetic unit in the Devanagari word. It is found that nearly 50% named entities used in India are compound of two or more individual named entities. For example, the named entity विजयराघवगढ़ (Vijayrāghavgarh - a place name) is formed using three named entities विजय (Vijay), राघव (Rāghav) and गढ़ (garh) respectively. For the one akshara, two aksharas and three aksharas named entities, transliteration is quite simple. As the length of named entity increases, the segmentation becomes important to find out the number of words used to form the named entity in order to separate the rhythms within it and in turn number of phonetic units in each rhythm. Following is the analysis based on the number of aksharas in the Hindi and Marathi named entities.

Most of the four aksharas named entities (denoted by NEs) are formed with the combination of two different words. Table 1 shows the possible combinations of phonetic segments from pronunciation point of view. In four aksharas word, if it is made up of any above combination, finding the boundaries of these combinations is important from transliteration point of view. It confirms that there are always minimum two segments in the four syllables word.

Table 1. Segmentations of Devanagari NEs consisting Four Aksharas

Named Entity	Segmentation
श्रीवर्धन(Shriwardhan)	श्री + वर्धन (Shrī + wardhan)
रामचंद्र(Rāmchandrā)	राम + चंद्र (Rām + chandrā)
धवलश्री(Dhawalshrī)	धवल + श्री (Dhawal + shrī)

The five aksharas named entities are formed with the combination of two different words. Table 2 shows the possible combinations of phonetic segments from pronunciation point of view. In five aksharas word, if it is made up of any above combination, finding the break point whether it occurs at second or third syllable is important. It confirms that the five aksharas named entity always consists of minimum two segments.

Table 2. Segmentations of Devanagari NEs consisting Five Aksharas

Named Entity	Segmentation
भानुप्रताप(Bhānuprātāp)	भानु + प्रताप (Bhānu + prātāp)
माणिकराव(Mānikrāo)	माणिक + राव (Mānik + rāo)
श्रीनारायण(Shrinārāyan)	श्री + नारायण (Shrī + nārāyan)

The six aksharas named entities are formed with the combination of two or three different words. Table 3 shows the possible combinations of phonetic segments from pronunciation point of view. In six aksharas word, if it is made up of any above combination, finding the break point whether it occurs at second, third, fourth or fifth syllable is important. It confirms that the six aksharas named entity always consists of minimum two segments.

Table 3. Segmentations of Devanagari NEs consisting Six Aksharas

Named Entity	Segmentation
करमरकर(Karmarkar)	कर + मर + कर (Kar + mar + kar)
प्रेमनारायण(Premnārāyan)	प्रेम + नारायण (Prem + nārāyan)
कमलकीशोर(Kamalkishor)	कमल + कीशोर (Kamal + kīshor)
जवाहरलाल(Jawāharlāl)	जवाहर + लाल (Jawāhar + lāl)

Most of the seven aksharas words are formed with the combination of two or three. In seven aksharas words, there can be two or three segments. As there are two or segments in the word finding the boundaries of these words is difficult task. It confirms that there are always two or more rhythms in the seven aksharas word. Table 4 shows the possible combinations of phonetic segments from pronunciation point of view.

Table 4. Segmentations of Devanagari NEs consisting Seven Aksharas

Named Entity	Segmentation
राजगुरुनगर (Rājgurunagar)	राज + गुरु + नगर (Rāj + guru +nagar)
गुरसहायगंज (Gursahāyganj)	गुर + सहाय + गंज (Gur + sahāy + ganj)
कंचनपुरकर (Kanchanpurkar)	कंचन + पुर + कर (Kanchan + pur+ kar)
मुरलीमनोहर (Muralimanohar)	मुरली + मनोहर (Murali + manohar)
नारायनस्वरुप (Nārāyanswarup)	नारायन + स्वरुप (Nārāyan + swarup)
गिरिराजकिशोर (Girirājkishor)	गिरिराज + किशोर (Girirāj + kishor)

Most of the eight aksharas words are formed with the combination of two or three words. In eight aksharas words, as there are two segments in the word, there has to be two or three rhythmic units. Table 5 shows the possible combinations of phonetic segments from pronunciation point of view. This analysis is useful to find out the number of segments and int turn number syllabic units commonly appear in the multi word named entities. This statistic clearly shows that the size of the n-gram in Hindi named entities is either two or three in most of the cases.

Table 5. Segmentations of Devanagari NEs consisting Eight Aksharas

Named Entity	Segmentation
विजयराघवगढ़ (Vijayrāghavgarh)	विजय + राघव + गढ़ (Vijay + rāghav + garh)
नांदुरखंदरमाळ (Nāndurkhandarmāl)	नांदुर + खंदर + माळ (Nāndur + khandar + māl)
नारायणगावकर (Nārāyangāonkar)	नारायण + गाव + कर (Nārāyan + gāon + kar)
त्रिभुवननारायण (Tribhuvannārāyan)	त्रिभुवन + नारायण (Tribhuvan + nārāyan)

4.2 SYLLABIFICATION

Syllabification is the process of dividing named entity written in Devanagari script into fundamental units called aksharas or syllabic unit. In our case study the one akshara in Devanagari it taken as one syllabic unit in English. It is needed to obtain syllabic unit alignment of source language named entity to target language named entity. This syllabification is useful to retain the phonemic features of the source language Hindi into transliterated form of English. It is also useful to train the input data according to the n-grams of the source language. It is to be noted that the length of the transliterated n-gram from Hindi to English will always differ. The syllabication of source language Hindi and its equivalent into target language English is depicted in table 6.

Table 6. Syllabification Format

Source Language(Hindi)	Target Language(English)
[ओ] [म]	[o] [m]
[ऐ] [श्व] [र्या]	[ai] [shwa] [rya]
[म] [हा] [रा] [ष्ट्र]	[ma] [hā] [rā] [shtra]
[ओं] [का] [रे] [श्व] [र]	[om] [kā] [re] [shwa] [r]
[अ] [ब्दु] [ल्ला] [ह] [गं] [ज]	[a] [bdu] [llā] [h] [gan] [j]
[नि] [रं] [ज] [न] [कु] [मा] [र]	[ni] [ran][ja] [n] [ku] [mā] [r]

5. IMPLEMENTATION

Our aim is to investigate the problem of machine transliteration where given a named entity in Hindi using Devanagari script need to be transliterated in English using CRF as a statistical probability tool and n-gram as a features set. The task is to generate a valid English language transliteration for the Hindi language as shown below.

$$X_1 \rightarrow Y_1$$
$$X_2 \rightarrow Y_2$$
$$\cdots$$
$$X_n \rightarrow Y_n$$

Here, X_i represents Hindi syllabic unit and Y_i represents English syllabic unit as shown in table 6. A conditional random field can be viewed as an undirected graphical model or Markov random field, globally conditioned on X, the

random variable representing observation sequence. For example, the named entity "माणिक" (mānik) a person name can be represented as shown in figure 3.

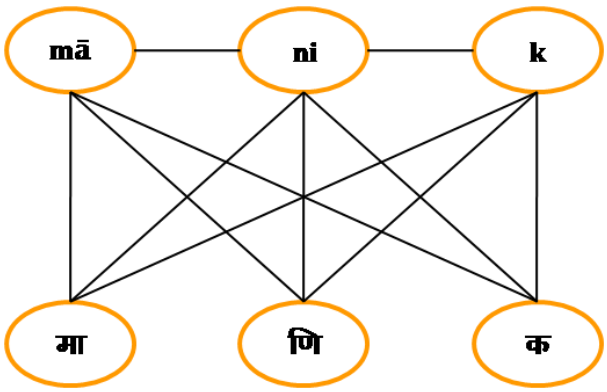

Figure 3. CRF Model Example

System implementation can be divided into two stages as follow:

5.1 TRAINING PHASE

Training phase requires two things, one is training data and other is features on which the data is to be trained. Parallel data obtained during syllabification is arranged in the CRF++ required format and then n-gram features are used to train this data.

Following templates are used to train the data for uni-grams, bi-grams and tri-grams.

U01:%x[0,1] is current observation under context.

U02:%x[1,1] is the observation next to the current observation under context.

U03:%x[2,1] is next to next observation to the current observation under context.

For training system on Unigram feature, only template U01 is used.

For example, if input from figure 2 is taken into consideration then the training data would look like as follows.

→	मा	मा	mā
	णि	णि	ni
	क	क	k

Here, if the first input token is taken into consideration, the output tag for 'मा' will depend only on current input which is 'मा', whereas for bigram feature U02 is used as template along with U01. In this case the output tag depends not only on current input 'मा' but also depends on next input which is 'णि'. In case of tri-gram, the output tag for 'मा' becomes dependent on both 'णि' and 'क' along with 'मा'.

Use of bi-gram feature can be elaborated in case of tagging the Hindi akshara 'व'. The Hindi akshara 'व' can be transliterated as 'v', 'va', 'w', 'wa', 'o' or 'on' depending on the context of input named entity. If a tag 'व' occurs at the end of name entity, for example, 'राजनंदगाव' (Rājnandgāon) which is place name, then most of the time it is to be tagged with 'on' otherwise for personal nouns it is normally tagged as 'v' or 'va' ('v' in झवर and 'va' in वरद). Using same analogy, other such examples can be trained using this feature and proper training data.

For trigram feature, a combination of U01, U02 and U03 is used as feature. Training of this feature can be elaborated using commonly used suffix 'नगर' (nagar) in most of the place names. For example, the named entity 'करिमनगर' (Karimnagar) which is a place name, the trigram feature 'नगर' is used and नगर is tagged as 'nagar' whereas in case of 'गजपाठीनगरम' (Gajpāthinagaram) which is also a place name, a sequence 'नगर' need to be tagged as 'nagara'. This difference in tagging can be modeled using combination of unigram and bigram features.

5.2 TESTING AND PERFORMANCE

To calculate the performance of the proposed approach there was the need of bilingual corpus in Unicode format. There were no source available; hence the bilingual corpus of 7251 named entities is created from web resources and books [34-45].

The test data includes personal names, surnames, and city and village names. The following notation are used for the evaluation metrics

N: Total number of names in the test set

R_i: i-th reference name (input) in source language in the test set

$C_{i,k}$: k-th candidate transliteration (output) for i-th name in the test set ($1 < k < 7$)

Ki: Number of candidate transliterations produced by a transliteration system

The commonly used performance evaluation parameter 'word accuracy' denoted by ACC is used to test the performance of our method. It measures correctness of the first transliteration candidate in the candidate list produced by a transliteration system [100]. ACC = 1 means that all top candidates are correct transliterations i.e. they match one of the references, and ACC = 0 means that none of the top candidates are correct [46].

$$ACC = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1 \text{ if } \exists Ri : Ri = C_{i1} \\ 0 \text{ Otherwise} \end{cases} \quad \textbf{(4)}$$

The performance ACC using equation (4) for uni-gram, bi-gram and tri-gram is shown in table 7.

Table 7. Performance

n-gram size	No of Records	Correct Match	Incorrect Match	ACC (Top-1)
Uni	7251	4714	2537	65.01%
Bi	7251	6221	1030	85.79%
Tri	7251	6090	1131	83.98%

Figure 4 shows the comparison based on the n-gram's size. It is very difficult to provide the comparative statements as no common bilingual corpus is used by others. Our results need not to be compared with the corpus as the records are created by user inputs.

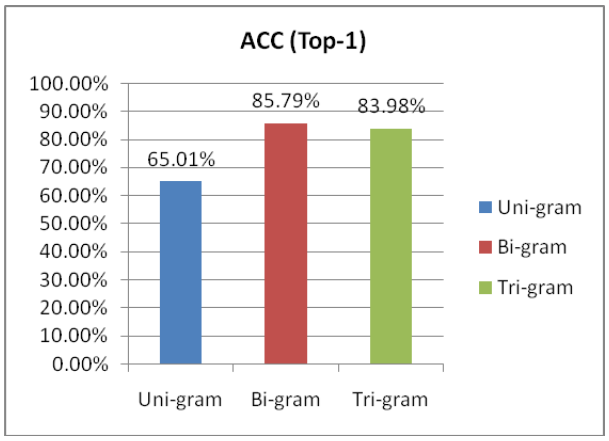

Figure 4. Comparison Based on n-gram size

6. CONCLUSION

In this approach, we presented machine transliteration of named entities for Hindi-English language pair using CRF as a statistical probability tool and n-gram as feature set. As the CRF calculates the probabilities over the entire input sequences, this approach is very good for the named entities of longer length. We have received very good accuracy 85.79% for the bi-grams of source language Hindi. As this approach is based on statistical probability, the results are always dependent on the training data size. The results for tri-gram are expected more than the bi-gram as per the literature review carried out by us but it may not have happened due to the inadequacy of training data. It has been observed that CRF is well suited for the Indian languages, as most of the named entities are made up of multiple smaller named entities.

7. ACKNOWLEDGMENTS

We express our gratitude to Dr. Jalnekar R M, Director and Prof. Haribhau Phakatkar, Dean Administration of Vishwakarma Institute of Technology for their continuous encouragement to carry out the research in the area of Machine Transliteration. We also thank to Prof. S. B. Karthick, Dr. S. N. Mali, Dr. Mansi Patwardhan, Prof. N. Z. Tarapore, Prof. M. M. Kulkarni, Prof. S. R. Bandewar, Dr. S.V. Joshi, Dr. Mukund Nalawade, Prof. Nitin Sahasrabuddhe and Prof. A. M. Kulkarni for their valuable guidance of Sanskrit and Hindi linguistics. Special thanks to Nandkishor Janardan Gaigol who has helped in segmenting around 3000 named entities in Hindi.

8. REFERENCES

[1] Ankit Aggarwal, Transliteration involving English and Hindi languages using syllabification approach, Thesis, Indian Institute of Technology, Bombay, Mumbai, 2009

[2] Haizhou Li, A Kumaran, Vladimir Pervouchine and Min Zhang, Report of NEWS 2009 Machine transliteration shared task, named entities workshop: shared task on transliteration, Singapore, pp. 1-18, 2009

[3] Darvinder kaur, Vishal Gupta, A survey of named entity recognition in English and other Indian languages, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 6, pp. 239-245, November 2010

[4] Karimi S, Scholer F, and Turpin, Machine transliteration survey. ACM Computing Surveys, Vol. 43, No. 3, Article 17, pp.1-46, April 2011.

[5] Arbabi M, Fischthal S M, Cheng V C and Bart E, Algorithms for Arabic name transliteration, IBM Journal of Research and Development. pp. 183-194, 1994

[6] Knight Kevin and Graehl Jonathan, Machine transliteration. In proceedings of the 35th annual meetings of the Association for Computational Linguistics, pp. 128-135, 1998

[7] Stalls Bonnie Glover and Kevin Knight, Translating names and technical terms in Arabic text. 1998

[8] Al-Onaizan Y, Knight K, Machine translation of names in Arabic text. Proceedings of the ACL conference workshop on computational approaches to Semitic languages. 2002

[9] Nasreen Abdul Jaleel and Leah S. Larkey, Statistical transliteration for English-Arabic cross language information retrieval. In Proceedings of the 12th international conference on information and knowledge management. pp: 139 – 146, 2003

[10] K Knight, J. Graehl, Machine transliteration , Computational Linguist, pp.128–135, 1997

[11] S. Y. Jung,, S. Hong, S., E. Paek,. English to Korean transliteration model of extended Markov window, In Proceedings of the 18th Conference on Computational Linguistics, pp.383–389, 2003

[12] R.K. Joshi, K. Shroff , S. P. Mudur, A Phonemic Code Based Scheme for Effective Processing of Indian Languages 23rd Internationalization and Unicode Conference, Prague, Czech Republic, 1 March 2003.

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

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

[15] R Sproat. Brahmi scripts, In Constraints on Spelling Changes: Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands, 2002.

[16] R. Sproat, A formal computational analysis of Indic scripts, In International Symposium on Indic Scripts: Past and Future, Tokyo, Dec. 2003.

[17] R. Sproat, A computational theory of writing systems, In Constraints on Spelling Changes: Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands, 2004.

[18] M. Kopytonenko, K. Lyytinen, and T. Krkkinen, "Comparison of phonological representations for the grapheme-to-phoneme mapping", In *Constraints* on Spelling Changes: Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands, 2006.

[19] Ganesh S, Harsha S, Pingali P, and Verma V, Statistical transliteration for cross language information retrieval using HMM alignment and CRF. In Proceedings of the

Workshop on CLIA, Addressing the Needs of Multilingual Societies, 2008

[20] Sujan Kumar Saha, Partha Sarathi Ghosh, Sudeshna Sarkar, and Pabitra Mitra, Named entity recognition in Hindi using maximum entropy and transliteration, 2008

[21] A Ekbal and S. Bandyopadhyay, A hidden Markov model based named entity recognition system: Bengali and Hindi as case studies, Proceedings of 2nd International conference in Pattern Recognition and Machine Intelligence, Kolkata, India, pp. 545–552, 2007

[22] A Ekbal and S. Bandyopadhyay, Bengali named entity recognition using support vector machine, in Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages, Hyderabad, India, pp. 51–58, January 2008

[23] A Ekbal and S. Bandyopadhyay, Development of Bengali named entity tagged corpus and its use in NER system, in Proceedings of the 6th Workshop on Asian Language Resources, 2008.

[24] A Ekbal and S. Bandyopadhyay, A web-based Bengali news corpus for named entity recognition, Language Resources & Evaluation, vol. 42, pp. 173–182, 2008.

[25] A Ekbal and S. Bandyopadhyay, Improving the performance of a NER system by post-processing and voting, in Proceedings of Joint IAPR International Workshop on Structural Syntactic and Statistical Pattern Recognition, Orlando, Florida, pp. 831–841, 2008

[26] A Ekbal and S. Bandyopadhyay, Bengali Named Entity Recognition using Classifier Combination, in Proceedings of Seventh International Conference on Advances in Pattern Recognition, pp. 259–262, 2009

[27] A Ekbal and S. Bandyopadhyay, Voted NER system using appropriate unlabelled data, in Proceedings of the Named Entities Workshop, ACL-IJCNLP 2009,

[28] A Ekbal and S. Bandyopadhyay, Named entity recognition using appropriate unlabeled data, post-processing and voting. In Informatica, Volume (34), No. 1, pp. 55-76, 2010.

[29] Manoj K. Chinnakotla, Om P. Damani, and Avijit Satoskar, Transliteration for Resource-Scarce Languages, ACM Trans. Asian Lang. Inform. Process. 9, 4, Article 14, pp 1-30, December 2010

[30] Jong-Hoon Oh, Kiyotaka Uchimoto, and Kentaro Torisawa, Machine transliteration using target-language grapheme and phoneme: Multi-engine transliteration approach, Proceedings of the Named Entities Workshop, ACL-IJCNLP Suntec, Singapore,AFNLP, pp. 36–39, August 2009

[31] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data,. In International Conference on Machine Learning, 2001.

[32] Hanna M. Wallach, Conditional Random Fields: An introduction, University of Pennsylvania CIS Technical Report MS-CIS-04-21, February , 2004

[33] Charles Sutton and Andrew McCallum, An Introduction to conditional random fields for relational learning, University of Massachusetts, USA

[34] http://www.whereincity.com/babynames

[35] http://en.wikipedia.org/wiki/list_of_cities_in_India

[36] http://www.indianchild.com/

[37] http://encyclopedia.thefreedictionary.com/

[38] Road Atlas Rajasthan – by Government of India, 2008

[39] Road Atlas Utter Pradesh – by Government of India, 2008

[40] Road Atlas Jharkhand – by Government of India, 2008

[41] Road Atlas Bihar – by Government of India, 2008

[42] Road Atlas Madya Pradesh – by Government of India, 2008

[43] Road Atlas Maharashtra – by Government of India, 2008

[44] Tourist Guide India - by Government of India, 2008

[45] Tourist Guide Maharashtra - by Government of India, 2008

[46] Haizhou Li, A Kumaran, Vladimir Pervouchine and Min Zhang, Report of NEWS 2009 Machine Transliteration Shared Task, ACL-IJCNLP, pp. 1-19, 2009