Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Kumar Saha, Sujan; Sarathi Ghosh, Partha; Sarkar, Sudeshna; Mitra, Pabitra

Named Entity Recognition in Hindi using Maximum Entropy and Transliteration

Polibits, vol. 38, 2008

Instituto Politécnico Nacional

Distrito Federal, México

Named Entity Recognition in Hindi using Maximum Entropy and Transliteration

Sujan Kumar Saha, Partha Sarathi Ghosh, Sudeshna Sarkar, and Pabitra Mitra

***Abstract*—Named entities are perhaps the most important indexing element in text for most of the information extraction and mining tasks. Construction of a Named Entity Recognition (NER) system becomes challenging if proper resources are not available. Gazetteer lists are often used for the development of NER systems. In many resource-poor languages gazetteer lists of proper size are not available, but sometimes relevant lists are available in English. Proper transliteration makes the English lists useful in the NER tasks for such languages. In this paper, we have described a Maximum Entropy based NER system for Hindi. We have explored different features applicable for the Hindi NER task. We have incorporated some gazetteer lists in the system to increase the performance of the system. These lists are collected from the web and are in English. To make these English lists useful in the Hindi NER task, we have proposed a two-phase transliteration methodology. A considerable amount of performance improvement is observed after using the transliteration based gazetteer lists in the system. The proposed transliteration based gazetteer preparation methodology is also applicable for other languages. Apart from Hindi, we have applied the transliteration approach in Bengali NER task and also achieved performance improvement.**

***Index Terms*—Gazetteer list preparation, named entity recognition, natural language processing, transliteration.**

I. Introduction

Named entity recognition is a subtask of information extraction that seeks to locate and classify the proper names in a text. NER systems are extremely useful in many Natural Language Processing (NLP) applications such as question answering, machine translation, information extraction and so on. NER systems have been developed for resource-rich languages like English with very high accuracies. But construction of an NER system for a resource-poor language is very challenging due to unavailability of proper resources.

English is resource-rich language containing lots of resources for NER and other NLP tasks. Some of the resources of English language can be used to develop NER system for a resource-poor language. Also English is used widely in many countries in the world. In India, although there are several regional languages like Bengali, Hindi, Tamil, Telugu etc., English is widely used (also as subsidiary official language). Use of the Indian languages in the web is very little compared to English. So, there are a lot of resources on the web, which are helpful in Indian language NLP tasks, but they are available in English. For example, we found several relevant name lists on the web which are useful in Hindi NER task, but these are in English. It is possible to use these English resources if a good transliteration system is available.

Transliteration is the practice of transcribing a word or text in one writing system into another. Technically most transliterations map the letters of the source script to letters pronounced similarly in the goal script. Direct transliteration from English to an Indian language is a difficult task. As our primary objective is to make the available English gazetteer lists useful for the Hindi NER task, we propose a two-phase transliteration, which is capable to do that.

The transliteration module uses an intermediate alphabet, which is designed by preserving the phonetic properties. The English names in the name lists are transliterated to the intermediate alphabet. A Hindi word, when it needs to be checked whether it belongs to a gazetteer list, is also transliterated into the intermediate alphabet. For an English-Hindi word pair, if their transliterated intermediate alphabet strings are the same, then we conclude that the English word is the transliteration of the Hindi word.

In this paper, we have identified suitable features for Hindi NER task. These features are used to develop a Maximum Entropy (MaxEnt) based Hindi NER system. The highest F-value achieved by the MaxEnt based system is 75.89. Then the transliteration based gazetteer lists are incorporated in the system and F-value is increased to 81.12. The improvement in accuracy demonstrates the effectiveness of the proposed transliteration approach.

The proposed transliteration module is applicable to other languages also. We have chosen another language Bengali and applied the transliteration approach for using the English gazetteers in Bengali NER task. Also in Bengali, the addition of the transliteration based gazetteer lists increases the accuracy.

The paper is structured as follows. Varios NER techniques and transliteration systems for different languages are discussed in Section II. In Section III, the architecture of the MaxEnt based Hindi NER system is presented. Then two-

Manuscript received July 10, 2008. Manuscript accepted for publication October 22, 2008.

Sujan Kumar Saha is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: sujan.kr.saha@gmail.com).

Partha Sarathi Ghosh is with HCL Technologies, Bangalore, India (email: partha.silicon@gmail.com).

Sudeshna Sarkar is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: shudeshna@gmail.com).

Pabitra Mitra is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: pabitra@gmail.com).

phase transliteration system is discussed in Section IV. In the next section, the prepared gazetteers and the corresponding experimental results are discussed. The experiments on Bengali NER are summarized in Section VI. Section VII presents the overall discussion. Finally Section VIII concludes the paper.

II. Previous Work

There are a variety of techniques for NER. Two broadly classified approaches to NER are:

- Linguistic approach and
- Machine learning based approach.

The linguistic approach is the classical approach to NER. It typically uses rules manually written by linguists. Though it requires a lot of work by domain experts, a NER system based on manual rules may provide very high accuracy. There are several rule-based NER systems, containing mainly lexicalized grammar, gazetteer lists, and list of trigger words, which are capable of providing F-value of 88-92 for English [9], [13], [18].

The main disadvantages of these rule-based techniques are: they require huge experience and grammatical knowledge on the particular language or domain; the development is generally time-consuming and sometimes changes in the system may be hard to accommodate. Also, these systems are not transferable, which means that one rule-based NER system made for a particular language or domain, cannot be used for other languages or domains.

The recent Machine Learning (ML) techniques make use of a large amount of annotated data to acquire high-level language knowledge. ML based techniques facilitate the development of recognizers in a very short time. Several ML techniques have been successfully used for the NER task. Here we mention a few NER systems that have used ML techniques.

'Identifinder' is one of the first generation ML based NER systems which used Hidden Markov Model (HMM) [2]. By using mainly capital letter and digit information, this system achieved F-value of 87.6 on English. Borthwick used MaxEnt in his NER system with lexical information, section information and dictionary features [3]. He had also shown that ML approaches can be combined with hand-coded systems to achieve better performance. He was able to develop a 92% accurate English NER system. Mikheev *et al*. has also developed a hybrid system containing statistical and hand coded system that achieved F-value of 93.39 [14].

Other ML approaches like Support Vector Machine (SVM), Conditional Random Field (CRF), Maximum Entropy Markov Model (MEMM) are also used in developing NER systems. Combinations of different ML approaches are also used. For example, we can mention a system developed by Srihari *et al*., which combined several modules, built by using MaxEnt, HMM and handcrafted rules, that achieved F-value of 93.5 [17].

The NER task for Hindi has been explored by Cucerzan and Yarowsky in their language independent NER which used morphological and contextual evidences [5]. They ran their experiments with 5 languages: Romanian, English, Greek, Turkish and Hindi. Among these, the accuracy for Hindi was the worst. For Hindi the system performance has F-value of 41.70 with very low recall 27.84% and about 85% precision. A successful Hindi NER system is developed by Li and McCallum using CRF with feature induction [12]. They automatically discovered relevant features by providing a large array of lexical tests and using feature induction to automatically construct the features that mostly increase conditional likelihood. In an effort to reduce overfitting, they used a combination of a Gaussian prior and early stopping. The training set consisted in 340 K words. Feature induction constructed 9,697 features from an original set of 152,189 atomic features; many are position-shifted but only about 1% are useful. Highest test set accuracy of their system is the F-value of 71.50. The MaxEnt based Hindi NER system developed by Saha *et al*. has achieved F-value of 80.01 [16]. The system has used word selection and word clustering based feature reduction techniques to achieve this result.

Transliteration is also a very important topic and several transliteration systems for different languages have been developed using different approaches. The basic approaches of transliteration are phoneme based or spelling-based. To mention a phoneme-based statistical transliteration system from Arabic to English is developed by Knight and Graehl [10]. This system used finite state transducer that implemented transformation rules to do back-transliteration. A spelling-based model that directly maps English letter sequences into Arabic letters is developed by Al-Onaizan and Knight [1]. There are several transliteration systems for English-Japanese [8], English-Chinese [11], English-Spanish [4] and many other languages to English.

But very few attempts were made to develop transliteration systems for Indian languages to English or other languages. We can mention a transliteration system for Bengali-English transliteration developed by Ekbal *et al*. [7]. They have proposed different models modifying the joint source channel model. In that system a Bengali string is divided into transliteration units containing a vowel modifier or *matra* at the end of each unit. Similarly, English string is also divided into units. Then various unigram, bigram or trigram models are defined depending on consideration of contexts of the units. Linguistic knowledge in the form of possible conjuncts and diphthongs in Bengali and their representations in English are also considered. This system is capable of transliterating mainly person names. The highest transliteration accuracy achieved by the system is 69.3% Word Agreement Ratio (WAR) for Bengali to English and 67.9% WAR for English to Bengali transliteration.

III. MAXENT BASED HINDI NER SYSTEM

We have used MaxEnt classifier to develop the system. Selection of an appropriate feature set is very important to train a ML based classifier. As language resources and tools are limited in Hindi, we have given the most importance to the features. MaxEnt model has the capability to use different features to compute the conditional probabilities.

In Hindi, there is no capitalization of letters to distinguish proper nouns from other nouns. Capitalization is a very important feature for English as most of the names are capitalized. Due to absence of the capitalization feature, Hindi NER task is difficult. Also, person names are more diverse in Indian languages; many common words are used as names.

In the following sections we discuss the features that we have identified and used to develop the Hindi NER system.

A. Feature Description

The features that we have identified for the Hindi NER task are:

- *Surrounding Words*

As the surrounding words are very important to recognize a NE, previous and next words of a particular word are used as features. As a feature, previous *m* words ($w_{i-m}...w_{i-1}$) to next *n* words ($w_{i+1}...w_{i+n}$) can be treated depending on the training data size, total number of candidate features etc. During experiment different combinations of previous four words to next four words are used as features. These features are multi-valued. For a particular word w_i, its previous word w_{i-1} can be any word in the vocabulary, which makes the feature space very high. Such high-dimensional features do not work well if amount of training data is not sufficient.

- *Binary Word Feature*

The multi-valued feature can be modified as a set of binary feature to reduce the feature space. Class specific lists are compiled taking the frequent words present in a particular position. For example, for the previous word of the *person* class, frequent words are collected in *PrevPerson* list. Such lists are compiled for each class and each position (previous *m* to next *n*). Now *C* binary features replace the word feature for a particular position, where *C* is the number of classes. The word in a particular position is checked whether it is in the corresponding position list for a class or not. Firstly we have prepared the lists blindly by taking the words occurring at least four times in a particular position corresponding to a class.

- *Context Lists*

The idea of binary word feature is used to define the class context features. Context words are defined as the frequent words present in a word window for a particular class. In our experiment we have listed all the frequent words present anywhere in $w_{i-3}...w_{i+3}$ window for a particular class. Then this list is manually edited to prepare the context word list for a class. For example, location context list contains *roda* (road), *rajdhani* (capital), *sthita* (located in), *jakar* (going to) etc. The feature is defined as, for a word w_i, if any of its surrounding words ($w_{i-3}...w_{i+3}$) is in a class context list then the corresponding class context feature is 1.

- *Named Entity Tags of Previous Words*

Named entity (NE) tags of the previous words ($t_{i-m}...t_{i-1}$) are used as feature. This feature is dynamic. The value of the feature for w_i is available after obtaining the NE tag of w_{i-1}.

- *First Word*

If the word is the first word of a sentence, then this feature is set to 1. Otherwise, it is set to 0.

- *Containing Digit*

If a word contains digit(s) then the binary feature *ContainsDigit* is set to 1.

- *Made up of 4 Digits*

For a word *w* if all the characters are digits and having only 4 digits in *w*, then the feature *fourDigit* is set to 1. This feature is helpful for identifying year. A little modification of the feature might give better result. As in our development, we are working in news domain, the years are limited to 1900-2100 in most cases. Then we have modified the feature as if it is a four-digit word and its value is between 1900 and 2100 then the feature value is 1.

- *Numerical Word*

If a word is a numerical word, i.e. it is a word denoting a number (e.g. *tin* (three), *char* (four) etc.) then the feature *NumWord* is set to 1.

- *Word Suffix*

Suffix information is useful to identify the named entities. This feature can be used in two ways. The first and naive one is that a fixed length word suffix of current and surrounding words can be treated as feature. During evaluation, it was observed that this feature is useful and able to increase the accuracy by a considerable amount. Still, better approach is to use suffix based binary feature. Variable length suffixes of a word can be matched with predefined lists of useful suffixes for different classes of NEs. Suffix list of locations is very useful since most of the location names in India end with a specific list of suffixes. Suffix list of locations contains 116 suffixes like, *bad, pur, puram, ganj, dihi* etc.

- *Word Prefix*

Prefix information of a word is also useful. A fixed length word prefix of current and surrounding words can be treated as feature.

- *Parts-of-Speech (POS) Information*

The POS of the current word and the surrounding words are important to recognize names. For this task we have used the POS tagger developed at IIT Kharagpur, India. The tagset of the tagger contains 28 tags. Firstly we have used the POS values of current and surrounding tokens as feature.

All 28 POS tags are not helpful in recognizing names. Nominal and postpositional tags are the most important in name finding in Hindi. Then we have modified the POS tagger to a coarse-grained POS tagger which has only three tags - nominal, postpositional (PSP) and others. These coarse grained POS values of current and surrounding tokens are more helpful for name recognition.

The POS information is also used in another way. Some binary features are defined using the POS information. For example, a binary feature *NominalPSP* is defined as following, if the current token is nominal and the next token is a PSP then the feature is set to 1, otherwise 0.

B. Maximum Entropy Based Model

MaxEnt is a flexible statistical model which assigns an output for each token based on its history and features. MaxEnt computes the probability *p(o|h)* for any *o* from the space of all possible outputs *O*, and for every *h* from the space of all possible histories *H*. A history is all the conditioning data that enables to assign probabilities to the space of output. In NER, history can be viewed as all information derivable from the training corpus relative to the current token w_i. The computation of *p(o|h)* depends on a set of features, which are helpful in making predictions about the output.

Given a set of features and a training corpus, the MaxEnt estimation process produces a model in which every feature f_i has a weight α_i. We can compute the conditional probability as [15]

$$p(o \mid h) = \frac{1}{Z(h)} \prod_i \alpha_i^{f_i(h,o)} \quad (1)$$

$$Z(h) = \sum_O \prod_i \alpha_i^{f_i(h,o)} \quad (2)$$

The probability is given by multiplying the weights of active features. The weight α_i is estimated by a procedure called Generalized Iterative Scaling (GIS) [6]. This method improves the estimation of weights iteratively. The MaxEnt estimation technique guarantees that, for every feature f_i, the expected value equals the empirical expectation in the training corpus.

For our development we have used a Java based open nlp MaxEnt toolkit[1] to get the probability values of a word belonging to each class. That is, given a sequence of words, the probability of each class is obtained for each word. To find the most probable tag corresponding to each word of a sequence, we can choose the tag having the highest class-conditional probability value.

Sometimes this method results in inadmissible assignment for tags belonging to the sequences that never happen. To eliminate these inadmissible sequences we have made some restrictions. Then we have used a beam search algorithm with beam length 3 with these restrictions. This algorithm finds the most probable tag sequence from the class conditional probability values.

C. Training Data

The training data used for this task contains of about *243 K* words with *16,482* NEs, which is collected from the popular daily Hindi newspaper "Dainik Jagaran". In this development, we have considered four types of NEs to recognize. These are *Person* (Per), *Location* (Loc), *Organization* (Org) and *Date*. To recognize entity boundaries, each name class *N* is subdivided into four sub-classes, i.e., *N_Begin*, *N_Continue*, *N_End*, and *N_Unique*. Hence, there are total *17* classes (4 name classes × 4 sub-classes + 1 not-name class). The corpus contains *6,298* Person, *4,696* Location, *3,652* Organization and *1,845* Date entities.

D. Evaluation

About 80 different experiments are conducted taking several combinations from the mentioned features to identify the best feature set for the NER task. We have evaluated the system using a blind test file of size 25 K words, which is totally different from the training file. The accuracies are measured in terms of F-measure, which is weighted harmonic mean of precision and recall. Precision is the percentage of the correct annotations and recall is the percentage of the total named entities that are successfully annotated. The general expression for measuring the F-value is: $F_\beta = ((1 + \beta^2)(precision \times recall)) / (\beta^2 \times precision + recall)$. Here the value of β is taken as 1.

First of all, we have used only the current and surrounding words as feature of MaxEnt. We have experimented with several combinations of previous 4 to next 4 words ($w_{i-4}...w_{i+4}$) to identify the best word-window. The results are shown in Table I.

TABLE I.
RESULTS (F-MEASURE) OF MAXENT BASED SYSTEM USING WORD FEATURES

Feature	Per	Loc	Org	Date	Total
w_i, w_{i-1}, w_{i+1}	61.36	68.29	52.12	88.9	67.26
w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}	64.10	67.81	58	92.30	**69.09**
w_i, w_{i-1}, w_{i-2}, w_{i-3}, w_{i+1}, w_{i+2}, w_{i+3}	60.42	67.81	51.48	90.18	66.84
w_i, w_{i-1}, w_{i-2}, w_{i-3}, w_{i-4}, w_{i+1}, w_{i+2}, w_{i+3}, w_{i+4}	58.42	64.12	47.97	84.69	61.27
w_i, w_{i-1}inList, w_{i-2}inList, w_{i+1}inList, w_{i+2}inList	65.37	70.33	47.37	83.72	66.17

From Table I we can observe that word window ($w_{i-2}...w_{i+2}$) gives the best result. When the window size is increased, the performance degrades. List based binary word features are not effective. In the table, the notation w_{i-n}*inList* is used to indicate binary word features for all classes for w_{i-n}. We have already mentioned that the binary word feature matches the word if it presents in a frequent word list which is formed from the training corpus. By analyzing the word lists we have observed that the lists do not contain all the words related to a class. For example, the word *'jakar'* (going to) in the next position helps to conclude that the current word has high probability to be a location name. But the word is *'jakar'* is not in the corresponding list because the word is not occurring in that particular position with high frequency in our training corpus. Manual editing of the lists might help the binary word feature to perform better.

Similar experiments are conducted to find the best feature set for the Hindi NER task. The features described earlier are applied separately or in combination to build the MaxEnt

[1] www.maxent.sourceforge.net

based model. In Table II we have summarized the results. Only the best values of each feature category are given in the table. This result is considered as the *baseline* in this study.

TABLE II.
RESULTS OF MAXENT BASED SYSTEM USING DIFFERENT FEATURES

Feature	***Per***	***Loc***	***Org***	***Date***	***Total***
words, previous NE tags	63.33	69.56	58.58	91.76	69.64
words, tags, prefix(≤4)	66.67	71	58.58	87.8	70.02
words, tags, suffix(≤4)	70	76.92	59.18	88.9	73.5
words, tags, suffix (≤4), prefix(≤4)	70.44	70.33	59.18	90.18	72.64
words, tags, digit information	62.94	69.56	50	91.76	67.63
words, tags, suffix (≤4), digit	70.44	76.92	60.44	93.02	74.51
words, tags, POS (28 tags)	66.67	72.84	60	88.9	71.22
words, tags, POS(coarse-grained)	69.62	80.74	58.7	91.76	75.22
words, tags, POS(coarse-grained), suffix (≤4), digit	72.23	78.1	62.37	93.02	75.67
words, tags, 'nominalPSP', suffix (≤4), digit	72.5	80.74	58.7	93.02	**75.89**

From the table we can observe that some of the features are able to improve the system accuracy separately, but when applied in combination with other features, they cause decreasing of the the accuracy. For example, with the information about the word and tag only we achieve F-value of 69.64. When suffix information is added, F-value is increased to 73.5 and when prefix information is added then F-value of 70.02 is achieved. But when both the suffix and prefix features are combined, then the F-value is 72.64. Prefix information increases the accuracy alone, but when combined with suffix information, it decreases the accuracy instead of increasing it. More complex features do not guarantee the better result. The best accuracy of the system is the F-value of 75.89, which is obtained by using current word, surrounding words (w_{i-1}, w_{i+1}), previous NE tags, suffix information (≤4), digit information (contains digit, four digit, numerical word) and the POS based binary feature *nominalPSP*. Here an interesting observation is, that the best feature set uses the word window (-1 +1), i.e. one previous word and one next word. Using the wider window reduces the performance, though in Table I it was found that window (-2 +2) performs best.

IV. GAZETTEER INFORMATION

Gazetteer lists or name dictionaries are helpful in NER. It is observed that a huge number of organization names end with some specific words like *Inc.*, *Corp.*, *Limited* etc. If all such words can be collected in a list then they can help to recognize the organization names. Again, it is very common that some designations like *prof.*, *minister* etc. and some other qualifiers like *Mr.*, *Dr.*, *Sri* etc. appear before the name of a person. A list containing all such words helps in person name identification. A surname list is also helpful for identifying person names. Similarly location list, organization list, first name list etc. are some helpful gazetteer lists.

Gazetteer lists are successfully used in many English NER systems. Borthwick's 'MENE' has used 8 dictionaries [3], which are: First names (1,245), Corporate names (10,300), Corporate names without suffix (10,300), Colleges and Universities (1,225), Corporate suffixes (244), Date and Time (51) etc. The numbers in parentheses indicate the size of the corresponding dictionaries. As another example, we can mention the hybrid system developed by Srihari *et al*. (2000). The gazetteer lists used in the system are: First name (8,000), Family name (14,000) and a large gazetteer of Locations (250,000). There are many other systems which have used name dictionaries to improve the accuracy.

Being influenced by these systems, we have decided to use gazetteer lists in our system. We have planned to use a few gazetteer lists like, person prefix, corporate suffix, surname, first name, location etc.

Initially we have attempted to prepare the gazetteers from the training corpus. Comparing with similar English dictionaries, it seems that prepared dictionaries might be sufficient for person prefix words, organization suffix words etc. but person first name list, location list etc. are not sufficient for the Hindi NER task. Then we have attempted to use the web sources for creating large gazetteer lists.

As our goal is to develop a NER system for Hindi, we are mainly interested in preparing gazetteers, which will contain mainly places in India, Indian first names and Indian surnames. For that purpose, we have collected the names from several websites. Mainly we have explored some Indian baby name websites to prepare the first name list. Also a lot of names of non-Indian famous personalities who are likely to appear in Indian news, collected from several sources, are added to the first name list. Similarly, we have prepared the location dictionary using Indian telephone directory, postal websites and the web encyclopedia 'wikipedia'. In Table III, we have mentioned the main sources from which we have collected the names.

TABLE III.
SOURCES OF GAZETTEER LISTS

Gazetteer	Sources
First name	http://hiren.info/indian-baby-names http://indiaexpress.com/specials/babynames http://www.modernindianbabynames.com/
Surname	http://surnamedirectory.com/surname-index.html http://en.wikipedia.org/wiki/Indian_name http://en.wikipedia.org/wiki/List_of_most_common_surnames
Location	http://indiavilas.com/indiainfo/pincodes.asp http://indiapost.gov.in http://maxmind.com/app/worldcities http://en.wikipedia.org/wiki

A. Transliteration

The transliteration from English to Hindi is very difficult. English alphabet contains 26 characters whereas the Hindi alphabet contains 52 characters. So the mapping is not trivial. We have already mentioned that Ekbal *et al*. [7] has

developed a transliteration system for Bengali. A similar approach can be used to develop a Hindi-English transliteration system. But it requires a bilingual transliteration corpus, which needs huge efforts to built, is unavailable to us. Also using this approach the word agreement ratio obtained is below 70%, which is not a good value for the task.

To make the transliteration process easier and more accurate, we propose a 2-phase transliteration module. As our goal is to make decision that a particular Hindi string is in English gazetteer or not, we need not transliterate the Hindi strings in English or English strings into Hindi. Our idea is to define an intermediate alphabet. Both the English and Hindi strings will be transliterated to the intermediate alphabet. For two English-Hindi string pair, if the intermediate alphabet is same then we can conclude that one string is the transliteration of the other.

First of all we need to decide the alphabet size of the intermediate state. When several persons write a Hindi name in English, all the English string may not be same. For example a Hindi name "*surabhii"* when written in English, may be written as several ways, like *surabhi, shurabhi, suravi, suravee, shuravi* etc. So, it is very difficult to transliterate properly. Preserving the phonetic properties we have defined our intermediate alphabet consisting of 34 characters. To indicate these 34 characters, we have given unique character-id to each character which ranges from 51# to 84#. As special characters and digits are very rare in person and location names, all the special characters are mapped to a single character with character-id 99# and all the digits are mapped to 98#.

B. English to Intermediate Alphabet Transliteration

For transliterating English strings into the intermediate alphabet, we have built a phonetic map table. This map table maps an English n-gram into an intermediate character. A few entities of the map table are shown in Table IV.

TABLE IV.
A PART OF THE MAP-TABLE

English	Intermediate	English	Intermediate
A	51#	EE, I	53#
OO, U	54#	B, W	55#
BH, V	56#	CH	57#
R, RH	76#	SH, S	77#

The procedure of transliteration is as follows.

Procedure 1: Transliteration English-Intermediate
Source string – English, Output String – Intermediate.

1. Scan the source string (S) from left to right.
2. Extract the first n-gram (G) from S. (n = 4)
3. Search it in the map-table.
4. If it is found, insert its corresponding intermediate state entity (I) into target string M. M → M + I.
   Remove G from S. S → S – G.
   Go to step 2.
5. Else, set n = n – 1.
   Go to step 3.

Using this procedure, English string '*surabhii'* will be transliterated to 77#54#76#51#56#53#. If we check the transliteration for '*shuravi'*, it is transliterated into intermediate string in the same manner.

C. Hindi to Intermediate Alphabet Transliteration

This is done in two steps. At the first step, the Hindi strings (which are in Unicode) are transliterated into *itrans*. *Itrans* is representation of Indian language alphabets in terms of ASCII. Since Indian text is composed of syllabic units rather than individual alphabetic letters, *itrans* uses combinations of two or more letters of English alphabet to represent an Indian language syllable. However, there are multiple sounds in Indian languages corresponding to the same English letter and not all Indian syllables can be represented by logical combinations of English alphabet. Hence, *itrans* uses some non-alphabetic special characters also in some of the syllables. The difficulty in converting the Unicode Hindi string to *itrans* is that the conversion mapping of Unicode to *itrans* is many to one. A map table[2], with some heuristic knowledge, is used for the transliteration. Our example Hindi word '*surabhii'* is converted into '*sUrabhI'* in *itrans*.

At the next step, the *itrans* string is transliterated into the intermediate alphabet using a similar procedure of transliteration. Here we use a similar map-table containing the mappings from *itrans* to intermediate alphabet. This procedure will transliterate the example *itrans* word '*sUrabhI'* to 77#54#76#51#56#53#.

D. Accuracy of the Transliteration System

The transliteration system is evaluated by using a bilingual corpus containing 1,070 English-Hindi word pairs most of which are names. 980 of them are transliterated correctly by the system. So, the system accuracy is 980×100/1070 = 91.59%.

This transliteration approach is applicable for some other languages also.

V. USE OF GAZETTEER LISTS IN MAXENT BASED HINDI NER

We have prepared the gazetteer lists directly from the corpus or from the web using the transliteration process discussed in the above section. The lists collected from the web are transliterated and stored in the intermediate form. One way of using the gazetteer information is to directly search a token if it is in the list. If it is present then we make the decision that the word belongs to that particular class. But this cannot resolve ambiguity as a particular token may present in more than one list and confusion arises. We have used the gazetteer information as a feature of MaxEnt. In the following we have described the prepared gazetteer lists and the corresponding features in details.

[2] www.aczoom.com/itrans

A. Gazetteer Lists

- *Month name, Days of the Week*

If the word is one of *January, February, . . ., December*, (*baishakh, jyashtha, . . ., chaitra* (month names of Hindi calendar)), then the feature *MonthName* is set to 1. If it is one of *Monday, Tuesday*, . . ., *Sunday (sombar, mangalbar, . . ., rabibar*,..) then the feature *DayWeek* is set to 1.

- *Corporate Suffix list*

Corporate Suffix List (CSL) contains most frequently occurring last words of organization names collected from the training data. CSL is made up of *limited*, *corp*., *inc*, *institute*, *university* etc. The size of the list is 92 entries. For a word w_i, if any of the words from w_{i+1} to w_{i+n} is in CSL, then a feature *CorpSuf* is set to 1.

- *Person Prefix List*

It contains the designations and qualifiers that occur before person names and are collected from the training data. Examples of some prefixes are, *sri* (Mr.), *kumari* (mrs.), *mantri* (minister), *adhyaksha* (chairman) etc. The list contains 123 prefixes.

Note that person prefix words are not the part of the person names, while corporate suffixes are part of the organization names. For a word w_i, if any of the words from w_{i-m} to w_{i-1} is in person prefix List, then a feature *PerPref* is set to 1.

- *Common Location*

This list contains the words denoting common locations. Common location words like *jila* (district), *nagar* (town/city), *roda* (road) etc. have high probability to occur at the end of a location name. 70 such words are collected in the Common Location List (CLL). Then the binary feature *ComLoc* is defined as, it takes value 1 for a word w_i if its next word presents in CLL.

- *Location List*

17,600 location names are gathered in the Location List (LL). LL is converted using the transliteration and stored in intermediate form. LL is processed into a list of unigrams (e.g., *Kolkata, Japan*) and bigrams (e.g., *New Delhi, New York*). The words are matched with unigrams and sequences of two consecutive words are matched against bigrams to get the feature value of the binary *LocList* feature.

- *First Name List*

This list contains 9,722 first names collected from the web. Most of the first names are of Indian origin. The feature *FirstName* is defined as, if the word w_i is in the list, then the feature is set to 1, otherwise 0.

- *Middle Name List*

A list is compiled containing the common middle names in India, for example, *kumar, chandra, nath, kanta* etc. This list contains 35 entries.

- *Surname List*

This is a very important list which contains surnames. As our objective is to develop a Hindi NER, we are most interested in Indian surnames. We have prepared the Surname List (SL) from different sources containing about 1,500 Indian surnames and 200 other surnames. A binary feature *SurName* is defined according to whether the word is in SL.

B. Evaluation

In Table V, we have shown the results of the NER system after incorporating the gazetteer lists. To observe the effectiveness of the prepared gazetteer lists in Hindi NER, we have added the lists with the baseline system.

TABLE V.
RESULTS OF MAXENT BASED SYSTEM USING GAZETTEER LISTS

Feature	Per	Loc	Org	Date	Total
Baseline: words, tags, suffix (≤4)	70	76.92	59.18	88.9	73.5
words, tags, suffix, CorpSuf	70	78.1	72.3	88.9	76.92
words, tags, suffix, DayWeek, monthName,	70	76.92	59.18	95.83	74.16
words, tags, suffix, PersonPrefix	72.5	76.92	59.18	88.9	74.09
words, tags, suffix, SurName, PerPref, FirstName, MidleName	77.2	78.1	59.18	88.9	76.34
words, tags, suffix, LocList, ComLoc	70	82.81	61.05	88.9	75.41
words, tags, suffix, all gazetteers	75.86	81.29	74.8	95.83	80.2
Baseline: words, tags, nominalPSP, suffix (≤4), digit	72.5	80.74	58.7	93.02	75.89
words, tags, nominalPSP, suffix, digit, all gazetteers	77.2	82.81	76.35	95.83	**81.12**

To observe the changes in accuracy, we have selected two feature sets from the baseline system (as in Table II): {current word, surrounding words, previous NE tags, suffix≤4} and {current word, surrounding words, previous NE tags, suffix≤4, digit information, nominal PSP}. The first feature set achieves F-value of 73.5 and the second one achieves F-value of 75.89, which is the best baseline feature set.

After adding the gazetteer lists, F-value has increased to 80.2 for the first feature set and 81.12 for the second. Also from the table we observe that the addition of a gazetteer list for a particular class (C_j) mostly increases the accuracy of C_j. For example, when the person gazetteer lists (e.g. person prefix list, surname list, first name list etc.) are incorporated, F-value of the person class has increased to 77.2 from 70. Change in accuracy of the other classes is minor. The highest F-value achieved by the developed Hindi NER system is 81.12.

VI. EXPERIMENTS ON BENGALI NER

The proposed two-phase transliteration approach is used successfully to make the English gazetteer lists useful in the Hindi NER task. The proposed approach is also applicable to other resource-poor languages. To study the effectiveness of the approach in another language we have chosen Bengali. As Hindi and Bengali alphabets are very similar, we needed a

little effort to transfer the transliteration module from Hindi to Bengali.

Our primary objective is not to develop a 'good' Bengali NER system, but to experiment the effectiveness of the transliteration approach in Bengali NER task. We first developed a Bengali NER system using a small training corpus which is used as baseline. Then the transliteration module is modified to make the collected English gazetteer lists useful for the Bengali. These gazetteer lists are incorporated in the system and the improvement in accuracy is observed.

A. Training Corpus

The training corpus used for the Bengali NER task is much smaller than the Hindi corpus. The corpus contains only 68 K words. Three named entity classes are considered: Person, Location and Organization. The corpus contains 1,240 person names, 1,475 location names and 490 organization names.

B. Transliteration Module

Collected English gazetteer lists are then transliterated into Bengali. English to intermediate alphabet transliteration of the gazetteer lists is already done during the experiments in Hindi. Using a Bengali map-table, the Bengali words are transliterated to *itrans*. We have already mentioned that the alphabets of Bengali and Hindi are similar, so the Hindi module for the transliteration from *itrans* to intermediate is used for Bengali without any modification.

The accuracy of the Bengali transliteration is measured using a smaller bilingual test corpus containing 400 word pairs. The accuracy of transliteration for Bengali is 89.3%.

C. Features for Bengali NER

The feature set used for the Bengali NER development is mentioned in the following.

- Surrounding words (two previous and two next),
- NE tags of previous words,
- Affix information (all affixes up to a fixed length and list based),
- Root information of the words,
- POS information.

Most of the features are used in similar ways as used in the Hindi NER task. The feature *root information* is not used in Hindi NER development, but it is very important in Bengali NER. In Bengali, several affixes are often added to the names inflecting them. For example, a person name "*Sachin*" is inflected in Bengali as, *sachinra* (plural, the group in which *Sachin* belongs to), *sachiner* (*of Sachin*), *sachinke* (*to Sachin*), *sachinda (brother Sachin*), *sachinbabu* (*Mr. Sachin*) etc. As these affixes are added to the names, sometimes identification of inflected names becomes very difficult. To identify the inflected names we have extracted the 'root' information of the words and used them as features of MaxEnt. In Hindi, such affixes generally present separately from the names as 'postpositions', so root information is not much useful.

D. Experimental Results

MaxEnt classifier is used for the experiments. The training corpus and the mentioned features are used to develop the baseline system. The system is evaluated using a test corpus containing 10 K words. The baseline system has achieved the highest F-value of 62.81. After that the transliteration based gazetteer lists are incorporated. Then F-value of the system has increased to 69.59. The results are summarized in Table VI.

TABLE VI.
RESULTS OF THE BENGALI NER SYSTEM

Feature	Per	Loc	Org	Total
words, tags	56.9	56.13	56.67	56.55
words, tags, affix	58.01	59.05	57.28	58.24
words, tags, affix, root information	62.21	60.46	57.94	60.6
words, tags, affix, root, POS information	64.39	62.5	60.2	62.81
words, tags, affix, root, POS information, all gazetteers	70.42	69.85	67.58	**69.59**

VII. DISCUSSION

Named entity recognition is an important task. ML based approach for NER task requires sufficient annotated data to build the system. Gazetteer lists are often used to increase the performance of a NER system. For resource-rich languages, such resources are available, but for resource-poor languages these resources are scarce. Useful gazetteer lists are not available in these languages, though sometimes they are available in other languages (like English). If such lists are transliterated from other language into the target language, they become useful. We have proposed a two-phase transliteration methodology for the task.

Direct transliteration is difficult, so we have proposed a two-phase transliteration. Here an intermediate alphabet is defined. The strings from both languages (say, Hindi and English) are transliterated into the intermediate alphabet to make the decision that a string (Hindi) is in the gazetteer lists (English) or not. The main advantages of the proposed approach are:

- This is a character-gram mapping based (using map-tables) approach, where no training data (bilingual corpora) is required.
- The approach is very simple and fast.
- This is easily transferable to other language.
- The accuracy of transliteration is high.

The disadvantages of the approach are:

- The English strings are not transliterated to the target language. Here only the decision is taken whether a target word (Hindi) is in the English name list or not.
- The module is specially built for the NER task. It is not widely applicable to other NLP tasks.

The accuracy of transliteration is 91.59% for Hindi and 89.3% for Bengali. The major cases where the transliteration

approach fails are, presence of homophones (pronounced similarly but one word is name but the other is not-name), word level changes (e.g., India is written as '*bharat*' in Indian languages, New Delhi as '*nayi dilli*'), dropping of internal vowels ('*surabhi*' is sometimes written/pronounced as '*surbhi*' – '*a*' is dropped) etc.

Suitable features are identified and MaxEnt is used to build the baseline NER systems for Hindi and Bengali using the identified features. Baseline accuracies for Hindi and Bengali are F-value of 75.89 and 62.81 respectively. A few gazetteer lists are collected from the web, which are in English, are incorporated in the system using the transliteration module and performance improvement is observed. F-values are increased to 81.12 for Hindi and 69.59 for Bengali. The accuracy for Bengali is much lower compared to Hindi because the training corpus size for Bengali is only 68 K words, whereas in Hindi the corpus contains 243 K words.

VIII. CONCLUSION

ML based approach requires annotated data and other resources to build a NER system. We have identified the suitable features for the Hindi NER task. We observed that some relevant gazetteer lists, which are very useful for improving the performance of the NER system, are available in English. To make the English name lists useful for Hindi, we have proposed a two-phase transliteration methodology. The available English gazetteer lists are used successfully in the Hindi NER system using the proposed transliteration approach. We have also examined the effectiveness of the transliteration approach on Bengali NER task.

Use of larger training data would increase the overall accuracy of the system. Also we hope that use of larger gazetteer lists will increase the accuracy of the system.

REFERENCES

[1] Al-Onaizan Y. and Knight K. 2002. Machine Transliteration of Names in Arabic Text. In: *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages.*

[2] Bikel D. M., Miller S, Schwartz R and Weischedel R. 1997. Nymble: A high performance learning name-finder. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 194-201.

[3] Borthwick A. 1999. A Maximum Entropy Approach to Named Entity Recognition. *Ph.D. thesis, Computer Science Department, New York University*.

[4] Crego J. M., Marino J. B. and Gispert A. 2005. Reordered Search and Tuple Unfolding for Ngram-based SMT. In: *Proceedings of the MT-Summit X,* Phuket, Thailand, pp. 283-289.

[5] Cucerzan S. and Yarowsky D. 1999. Language independent named entity recognition combining morphological and contextual evidence. In: *Proceedings of the Joint SIGDAT Conference on EMNLP and VLC 1999,* pp. 90-99.

[6] Darroch J. N. and Ratcliff D. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, pp. 43(5):1470-1480.

[7] Ekbal A., Naskar S. and Bandyopadhyay S. 2006. A Modified Joint Source Channel Model for Transliteration. In *Proceedings of the COLING/ACL 2006*, Australia, pp. 191-198.

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

[9] Grishman R. 1995. Where's the syntax? The New York University MUC-6 System. In: *Proceedings of the Sixth Message Understanding Conference*.

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

[11] Li H., Zhang M. and Su J. 2004. A Joint Source-Channel Model for Machine Transliteration. In: *Proceedings of the 42nd Annual Meeting of the ACL*, Barcelona, Spain, (2004), pp. 159-166.

[12] Li W. and McCallum A. 2003. Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. In: *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3): 290–294.

[13] McDonald D. 1996. Internal and external evidence in the identification and semantic categorization of proper names. In: *B.Boguraev and J. Pustejovsky (eds), Corpus Processing for Lexical Acquisition*, pp. 21-39.

[14] Mikheev A, Grover C. and Moens M. 1998. Description of the LTG system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference*.

[15] Pietra S. D., Pietra V. D. and Lafferty J. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 19(4): 380-393.

[16] Saha S. K., Mitra P. and Sarkar S. 2008. Word Clustering and Word Selection based Feature Reduction for MaxEnt based Hindi NER. In: *proceedings of ACL-08: HLT*, pp. 488-495.

[17] Srihari R., Niu C. and Li W. 2000. A Hybrid Approach for Named Entity and Sub-Type Tagging. In: *Proceedings of the sixth conference on applied natural language processing*.

[18] Wakao T., Gaizauskas R. and Wilks Y. 1996. Evaluation of an algorithm for the recognition and classification of proper names. In: *Proceedings of COLING-96*