Marathi/Hindi to English Transliteration of Named Entity using WEKA Tool

[1]Mayur Navsupe, [2]Manikrao Dhore
Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, India

***Abstract* - Phonetic property of a word is the property which describes the speech and pronunciation sound. Transliteration is a mapping of a word into one language from another language without losing its phonetic properties. Named entities (NE) are the words which are not found in the dictionary but they are used to support translation process. The process of transliteration of these named entity is always have been difficult. There are no predefined rules or regulation for transliteration which will not lead to loss of phonetic properties of named entities. People are writing different spellings at different places for the same name. This fact certainly affects the top-1 accuracy of the transliteration and in turn the transliteration process. We introduce the concept to use machine learning approach on NE data to solve this problem. We have used WEKA tool, a machine learning tool for training of data. We have used SMO classifier for the classification of NE data. We have applied set of linguistic rules on the predicted data for better results. Initial experiment shows use of machine learning tool and SMO is very helpful for transliteration process.**

***Key Words* - Transliteration, Named Entity, WEKA, SMO (Sequential minimal optimization), Linguistic.**

I. INTRODUCTION

Since the evolution of India, Hindi is the National and Official language in most of the state Governed states and in all the Centrally Governed States whereas Marathi is official language of Maharashtra state. But day by day English is also taking major role in official works as most of official works are getting digitalized and English is only global and standardised one. So, there is wastage of lots men work hours in simply translating the Marathi/Hindi Words (like Names, Surnames) into proper English spells. It is challenging to transliterate out of vocabulary words like identity names, names of places and technical terms occurring in the user input across languages with different characters (alphabets) and sounds. Hindi and Marathi to English named entity (henceforth denoted as NE) transliteration is quite difficult due to many factors such as difference in script of language, number of alphabets, capitalization of initial Characters, phonetic properties, length of character, number of valid transliterations.[1]

There are two types of transliteration model which are as follows

i. Grapheme Based: It uses orthographic process and directly associates the source language grapheme/character to respective target language grapheme/character.

ii. Phoneme Based: It uses phonetic process and thus converts source grapheme/character to source phoneme and then source phoneme to target grapheme/character.

There are two primary needs for transliteration. They are:
1. In India, huge majority of the population use their mother-tongue as the medium of communication.
2. In spite of globalization and wide-spread influence of the West in India, most of the people still prefer to use their mother-tongue.

Taking these two factors into consideration, we realize that for an Internet revolution to reach out to the majority of the population; it should be accessible in the language, which is known to the user.

Therefore, mere approach of machine translation will not be inadequate to fulfil the needs of a correct source language to target language without change in phonetic property.

II. RELATED WORK

In this section we provide the previous work done on the transliteration systems using machine learning techniques.

Antony P J, Ajith VP, and Soman KP of Amrita University, Coimbatore proposed three different approaches for English to Kannada Transliteration [2]. The first proposed transliteration model out of three is based on multiclass classification for which j48 decision tree classifier of WEKA was used for classification. They trained classifier with 40,000 Indian place names.

System is then tested with 1000 English names giving 81.25% Accuracy for top 1 result, 85.88% accuracy for top 2 results and 91.32% Accuracy for top 5 results. The second method addresses the problem of Transliterating English to Kannada language using Support Vector Machines. They used sequence labeling method to model the transliteration problem. The framework was based on data driven method and one to one mapping approach. The model is trained on 40,000 words containing Indian place names. System achieved exact Kannada transliterations for 87.28% of English names for top-5 results. Third English to Kannada transliteration system was developed using a publically available translation tool called Statistical Machine Translation. System was trained with 40,000 words containing Indian Places. System achieved exact Kannada transliterations for 89.27% of English names.

Table1. Transliteration achieved by different approaches.

Authors	**Year**	**Source and Target Languages.**	**Methodology**
Kumaran A and Tobias Kellner [3]	2007	English to Tamil transliteration system.	Machine transliteration framework based on a core algorithm modeled as a noisy channel.
Vijaya MS, Ajith VP, Shivapratap G, and Soman KP of Amrita University, Coimbatore.[4]	2008	English to Tamil Transliteration using machine learning approach.	Modeled as classification problem and trained using C4.5 decision tree classifier, in WEKA Environment.
Kang B. J. And Key-Sun Choi.[5]	2008	English to Korean automatic transliteration and back-transliteration by decision tree learning.	-Grapheme-based transliteration models -English grapheme to Korean grapheme conversion model based on decision trees.

III. WEKA TOOL AND SMO:

WEKA (Waikato Environment for Knowledge Analysis) is open source machine learning software based on Java and it is developed at the University of Waikato, New Zealand. WEKA tool contains variety of machine learning algorithms and supports several data mining tasks. WEKA tool have features like data pre-processing, classification, clustering and visualization. The machine learning algorithms can directly applied to user dataset or can be called from Java code.

(Sequential minimal optimization) SMO is an algorithm implemented in LIBSVM tool which is also present in WEKA tool. SMO is widely used to solve quadratic programming (QP) problem that arises during the training of support vector machine.

Consider a binary classification problem with a dataset (x_1, y_1), ..., (x_n, y_n), where x_i is an input vector and $y_i \in \{-1, +1\}$ is a binary label corresponding to it. A soft-margin support vector machine is trained by solving a quadratic programming problem, which is expressed in the dual form as follows:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j K(x_i, x_j)\alpha_i\alpha_j,$$

subject to:

$$0 \le \alpha_i \le C, \quad \text{for } i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} y_i\alpha_i = 0$$

where C is an SVM hyper parameter and $K(x_i, x_j)$ is the kernel function, both supplied by the user; and the variables α_i are Lagrange multipliers.

The SMO algorithm selects two α parameters, αi and αj and optimizes the objective value jointly for both these α's. Finally it adjusts the b parameter based on the new α's. This process is repeated until α's converge [6].

IV. PRIMITIVE SCRIPT:

Both Marathi and Hindi languages were originated from Devanagari script and because of this both languages share number of similar properties. Devanagari script is consists of 34 consonants, 7 loan consonant, 5 conjunct and 2 traditional signs. It also has 12 pure vowels, 2 and 1 loan vowels from Sanskrit and English respectively. Also each consonant has 14 variations combining with 14 vowels.

V. SYSTEM ARCHITECTURE

In this section we talk about the methodology that we have adapted to achieve transliteration of Marathi/Hindi named entity into English language. As given bellow, Figure 1 shows the system architecture of proposed system. This section also put light over the process through which raw data goes to be compatible for our proposed system. This section includes the details regarding training phase of data and information regarding testing phase.

The phases in which system architecture can be categorised are as follows.

1) Gathering of Training data
2) Training through WEKA tool
3) Testing of system

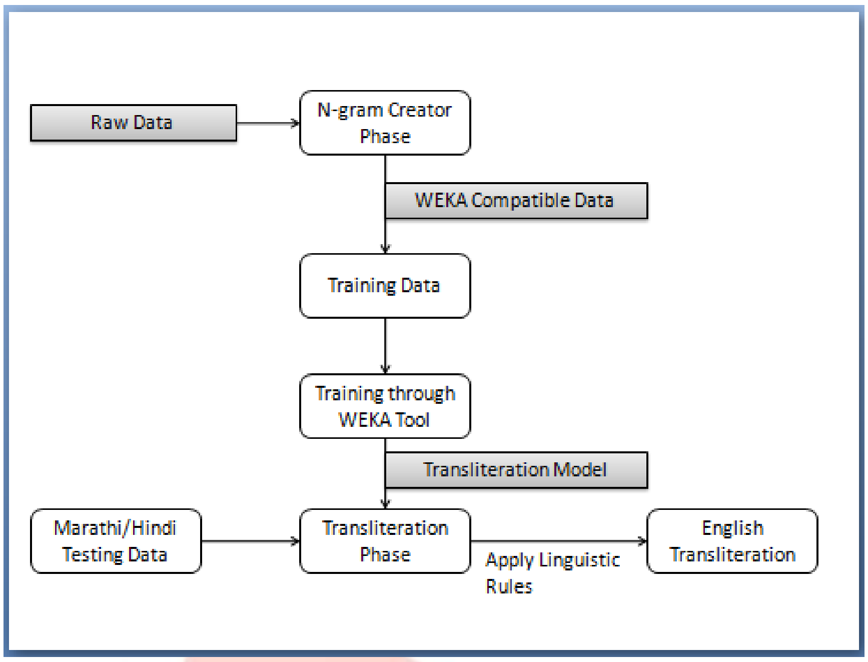

Figure 1: System Architecture

V.I Gathering Of Training Data

This section talk about the pre-processing of raw data into acceptable WEKA tool input format. Also about N-gram creator class which enhance the training data into more useful data. As the proposed methodology uses the phoneme of source input as one of the feature, representation of raw input in Devanagari needs to be done using the syllabic format of the source language. In this approach one syllabic unit of source language has treated as a one phonetic unit. The overall pre-processing have achieved by dividing task into following two sub modules.

· Syllabification
· Phonetic mapping

V.I.I Syllabification

In this phase we divide the NE's into segments of syllabic unit of source language.

These segments of NE, i.e, source transliterated unit (STU) are then mapped into translated target units (TTU). Syllabification of source NE has done with the help of Java 6 which has in built Hindi Locale support , Locale hindi = new Locale("hi","IN"). It works on the same principle as of Formation of Devanagari Phonetic Transliteration Units algorithm[7].

for example NE **मयुर नवसुपे** in Devanagari will be converted as

NE	STU's
मयुर	**म | यु | र**
नवसुपे	**न | व | सु | पे**

V.I.II Phonetic Mapping

After the conversion of NE into STU's, we have manually mapped it to Transliterated target units (TTU's) as shown bellow.

NE	STU's	TTU's
मयुर	[**म | यु | र**]	[ma | yu | r]
नवसुपे	[**न | व | सु | पे**]	[na | v | su | pe]

We have did this manually because when you take example in the case of /**व**/ which is pronounced as 'va' and 'v' so it can be mapped to 'v' and 'va' that's why It is difficult to decide which one to use in a automatic way. Hence we have done this phase manually.

V.I.III N-GRAM CREATOR

After phonetic mapping of NE's is done, it is then passed through the Java class written by us, which converts these NE into n-grams set. It creates the different combinations of STU's and TTU's gram wise. Following are the example of NE to different N-gram parallel STU's and corresponding TTU's.

1) [सु | रें | द्र] -> [su | ren | dra]

Unigram : Keeping single STU's and TTU's
STU: { सु , रें , द्र } TTU:{ su , ren , dra }

Bigram: Keeping two consecutive STU's and TTU's
STU: { सुरें , रेंद्र , सु , द्र } TTU:{ suren, rendra, su, dra }

Trigram: Keeping three consecutive STU's and TTU's
STU: { सुरेंद्र } TTU:{surendra}

2) [र | म | न | वी | र] -> [ra | ma | n | vee | r]

Unigram: Keeping single STU's and TTU's
STU: { र , म , न , वी , र }
TTU: { ra , ma , n , vee , r }

Bigram: Keeping two consecutive STU's and TTU's
STU: { रम , नवी , र , मन , वीर }
TTU: { rama, nvee, r, man, veer }

Trigram: Keeping three consecutive STU's and TTU's
STU: { रमन, वीर , मनवी, र, नवीर }
TTU: {raman,veer,manvee,r,nveer}

Fourgram: Keeping four consecutive STU's and TTU's
STU: { रमनवी,र , मनवीर }
TTU: {ramanvee,r,manveer}

Fivegram: Keeping five consecutive STU's and TTU's
STU: { रमनवीर }
TTU: {ramanveer}

V.I.IIII WEKA compatible data

After the data has processed through n-gram creator class, we have got output as STU's and corresponding TTU's separated by comma in a .csv extension file. This comma separated file contains Marathi/Hindi NE data in one column and corresponding transliterated English NE is next column. CSV file is then given to WEKA tool for further training of this data. For our system we have used 5000 NE data for training purpose.

V.II Training Phase

The processed data available in csv file is accepted by CSVLoader class of WEKA explorer tool. Then this data is trained with SMO classifier of weka with second degree Polynomial kernel. Output of this phase is a .model file which we have used as a predictor for this system.

Based on n-gram training data, polynomial kernel of SMO divides theses STU's and TTU's pattern to positive and negative space as follows.

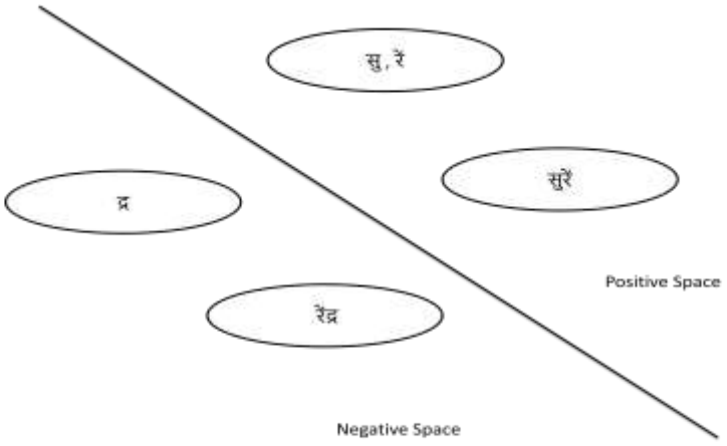

Figure 2: Hyperplane for NE “**सुरेंद्र**”

V.III Testing Phase

This phase takes two files as input. One is the model file generated during training phase and other is test file. The format of a test file is same as training file. It searches a particular pattern from a separating hyper plane. Each pattern has a score value either positive or negative which indicates the distance of class form a separating hyperplane in model file. If the given Hindi and Marathi input is in combination of the patterns which are there in model file, then generates the correct output. If the pattern is not found then it shows garbage output i.e. incorrect English transliterated output.

For example: **रमन** is given for prediction to model file with unigram method on test file then the output is predicted as follows

र -> ra
म ->ma
न ->n

V.IV Apply Linguistic Rules

In this step, we have applied set of linguistic rules based on the output of transliteration phase. Based on initial experiment, we have noticed there are very few “aa” in any named entity so we have eliminated it from the predicted output. As “गाव” is transliterated as “gav” but when used in the named entity like,”उचकगाव” it should transliterated as “goan” for “Uchakgaon”. Therefore for better accuracy, we have applied a rule for “गाव” as “goan” not “gav” or “gava” at the end of NE. Same case for “वाडी”, instead of “vadi” we have applied rule “wadi”.

VI. EXPERIMENTAL DETAILS

This section talk about the overall implementation details of Marathi/Hindi to English machine transliteration using WEKA tool.

VI.I Configuration

The implementation is done with the help of Netbeans development environment which has inbuilt support for JAVA programming. The GUI is created using Java swings .WEKA SMO is set with polynomial kernel of degree two. Output of training phase is a .model file which is saved and used as predictor in system. For testing, NE is divided into grams same as training data and predicted for each gram wise.

VI.II Testing and Result

The transliteration systems GUI is as shown in figure 2.

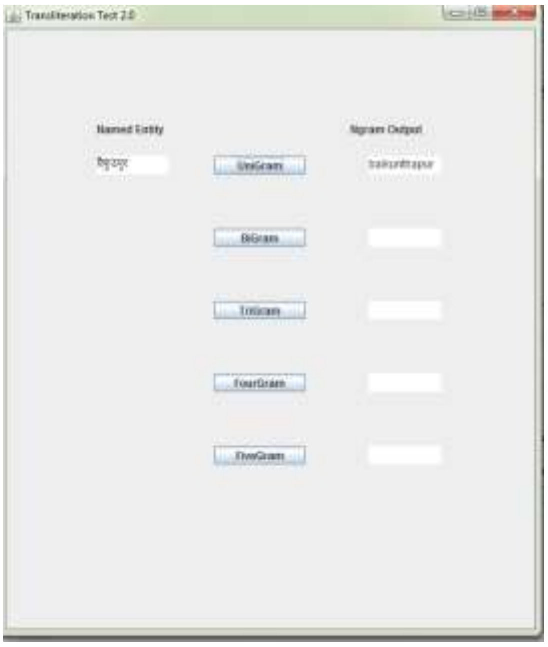

Figure 2: GUI of transliteration system.

Training file contains 5000 names of persons, places. This file is then given to WEKA tool and model file is saved. Test file consisting of 1000 NE is used for testing which given the results are shown in Table1.

Table 2: Test Result

	Bi-gram	**Tri-gram**	**Four-gram**	**Fivegram**
Number of NE trained	5000	5000	5000	5000
Number of NE tested	1000	1000	1000	1000
Number of correct NE	485	519	782	822
Accuracy	48.5	51.9	78.2	82.2

VI.III Result Analysis

The NEs with length two and length three has shown good accuracy for bigram and trigram respectively. From the result table 1, we have seen that the accuracy is almost same in case of fourgram and fivegram. We have trained 5000 NE using WEKA Toolkit and tested for 1000 NE data. When we have taken 1000 named entities as testing data, we have found maximum accuracy as 48.5% for bigram , 51.9% using trigram , 78.2% using fourgram and 82.2% using fivegram .

VII. CONCLUSION

In this paper, we put forward the Hindi/Marathi to English machine transliteration using WEKA tool. We have used SMO algorithm of WEKA tool as machine learning algorithm for classification purpose. As SMO creates the hyper plane using linear polynomial function, it is useful for multiclass classification. Accuracy has been gradually increased as the n-gram sized increased. Result analysis shows that 5-gram gives best result which is 82.2% for Hindi and Marathi to English named entity transliteration.

REFERENCE

[1] M. L. Dhore "Issues and Possible Solutions for Hindi and Marathi to English Machine Transliteration of Named Entities", ICRTET at SNJB's Late Sau. K. B. Jain College Of Engineering, Chandwad (2014)

[2] P. J. Antony, Ajith, V. P., and Soman, K. P., "Statistical method for English to Kannada transliteration", International Conference on Recent Trends in Business Administration and Information Processing, BAIP 2010. Springer, Trivandrum, Kerala, India, pp. 356–362 (2010),

[3] Kumaran, A., Kellner, Tobias: "A generic framework for machine transliteration." In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval , 2007, . pp. 721-722 (2007).

[4] Vijaya MS, Ajith VP, Shivapratap G, and Soman KP CEN, Amrita University, Coimbatore, India "English to Tamil Transliteration using WEKA" in International Journal of Recent Trends in Engineering, Vol. 1, No. 1(2009)

[5] Kang B.J, Key-Sun Choi,. "Automatic Transliteration and Back-transliteration by Decision Tree Learning", Proceedings of the Second International Conference on Language Resources and Evaluation(2000).

[6] Platt, John. “Fast Training of Support Vector Machines using Sequential Minimal Optimization “ in Advances in Kernel Methods – Support Vector Learning, B. Scholkopf, C. Burges, A. Smola, eds., MIT Press (1998).

[7] M L Dhore, P H Rathod. “Transliteration by orthography or phonology for hindi and marathi to english: case study” in International Journal on Natural Language Computing (ijnlc) vol. 2, no.5, (2013)