HINDI AND MARATHI TO ENGLISH MACHINE TRANSLITERATION USING SVM

P H Rathod[1], M L Dhore[2], R M Dhore[3]

[1,2]Department of Computer Engineering, Vishwakarma Institute of Technology, Pune
[1]pravin.rathod@vit.edu
[2]manikrao.dhore@vit.edu
[3]Pune Vidhyarthi Griha's College of Engineering and Technology, Pune
[3]ruchidhore93@gmail.com

ABSTRACT

Language transliteration is one of the important areas in NLP. Transliteration is very useful for converting the named entities (NEs) written in one script to another script in NLP applications like Cross Lingual Information Retrieval (CLIR), Multilingual Voice Chat Applications and Real Time Machine Translation (MT). The most important requirement of Transliteration system is to preserve the phonetic properties of source language after the transliteration in target language. In this paper, we have proposed the named entity transliteration for Hindi to English and Marathi to English language pairs using Support Vector Machine (SVM). In the proposed approach, the source named entity is segmented into transliteration units; hence transliteration problem can be viewed as sequence labeling problem. The classification of phonetic units is done by using the polynomial kernel function of Support Vector Machine (SVM). Proposed approach uses phonetic of the source language and n-gram as two features for transliteration.

KEYWORDS

Machine Transliteration, n-gram, Support Vector Machine, Syllabification

1. INTRODUCTION

As Internet users are growing day by day, it is logical to develop tools and applications to support Indian languages for them. It is challenging to transliterate out of vocabulary words like personal names, location names and technical terms occurring in the user input across languages with different characters (alphabets) and sounds. Transliteration is a mapping of a word from one language to another language without losing its phonetic properties [1]. Hindi and Marathi to English named entity (henceforth denoted as NE) transliteration is quite difficult due to many factors such as difference in writing script, number of alphabets, capitalization of initial characters, phonetic properties, length of character, number of valid transliterations and availability of the parallel corpus [2]. Formally, transliteration can be defined as the conversion of a given named entity in the source language i.e. a text string in the source writing system or orthography, to a name entity in the target language i.e. 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. If transliteration is taken as a sequence labeling problem, the transliteration can be defined as a task to generate a valid target language character/s or label sequence for the source language character/s or observation sequence as shown below.

$$X_1 \rightarrow Y_1, \quad X_2 \rightarrow Y_2, \quad \ldots \quad X_n \rightarrow Y_n$$

where X_i represents alphabet in Hindi or Marathi and Y_i represents English character.
For Indian languages, most methods incorporated are statistical in nature and most of the work is carried out for English to Indian languages. Very less amount of research is carried for Indian languages to English. Statistical methods do not require extensive knowledge of language, but they do need large amount of training data which runs parallel in both languages. The requirement of bilingual corpus is satisfied by using information sources like directories, government documents, voters' list, student's enrolment lists of schools and colleges etc.

This approach uses proper manual alignment method and two features as phonetic and n-gram to improve accuracy of existing transliteration system. The proposed method uses SVM which is one of the most efficient statistical supervised learning mechanisms to obtain the transliteration.
Initially broad survey of various methods used for Transliteration in Indian and Foreign languages is presented. Then a detailed analysis of our approach is given which concludes that SVM suits the most for the task of transliteration. Finally a transliteration system based on SVM is proposed and tested for Marathi and Hindi to English language pairs.

2. RELATED WORK

Existing approaches for machine transliterations are Grapheme-based and Phoneme-based. The grapheme based model assumes transliteration as an orthographic process and map the source language graphemes directly to the target language graphemes. Phoneme-based model treats transliteration as a phonetic process. In phoneme-based framework, transliteration is treated as a conversion from source grapheme to source phoneme followed by a conversion from source phoneme to target grapheme.

C-DAC (Centre for Development of Advanced Computing), NCST (National Centre for Software Technology) and Indictrans Team have played major role in the machine transliteration of Indian languages in India. The development of GIST (Graphics and Intelligence - Based Script Technology) was a major breakthrough in early 1980. C-DAC developed GIST card which was based on Indian Script Code for Information Interchange (ISCII). Another major development was UTF-8 Unicode based coding for Indian languages [3]. The third development (2003) was a phonemic code based scheme for effective processing of Indian languages which was used for transliterating telephone directory in Hindi, and voters' list [4]. The few other applications localised were Indian Railways Reservation Systems, Telephone Bills and Bilingual Telephone Directories.

Early work on transliteration is done by Arbabi in 1994. He combined neural networks with expert systems for Arabic-English language pair using phoneme-based model [5]. In 1997, Knight and Graehl suggested a five stage statistical model for back transliteration to recover the original English name from Japanese Katakana [6]. Stalls used the same method for back transliteration from Arabic to English language pair in 1998 [7]. In 2002, Al-Onaizan and Knight developed a simpler Arabic-English transliterator and evaluated how well their system can match a source spelling [8]. Their work inculcates an evaluation of the transliterations in terms of their reasonableness according to human judges. Work on CLIR for Indian Languages were done by Jaleel and Larkey which was analogous to their previous work on English-Arabic machine transliteration in 2003[9]. The approach was based on HMM using GIZA++. In 1997, Knight developed phoneme-based models, based on weighted finite state transducers (WFST). Jung used Markov window which considers transliteration as a phonetic process in 2003[10]. In 2005, OM transliteration scheme provided a common script representation for most of the Indian languages

[11]. Punjabi machine transliteration for Punjabi language from Shahmukhi to Gurmukhi used the rule-base approach for transliteration in 2006 [12]. During 2002 to 2004, Sproat discussed a formal computational analysis of Brahmi scripts [13-15]. Kopytonenko presented computational models to carry out grapheme-to-phoneme conversion in 2006 [16]. In 2008, Ganesh developed a statistical transliteration technique which is language independent. Their team chosen a statistical model for transliteration which was based on HMM alignment and CRFs [17]. Sujan Kumar Saha proposed a two-phase transliteration methodology in 2008. In 2009, Sumaja Loganathan R and Soman K P demonstrated method for English to Malayalam transliteration by using segmentation of the source into transliteration units and mapping the source language transliteration into the target language. In experimentation part they have used SVMTool for giving the training to corpus. SVM learning was done using linear kernel. They proved that learning time remains linear with respect to the number of examples [18]. Jong-Hoon Oh approach was based on two transliteration models [19]. They used three different machine learning approaches MIRA, MEM and CRF for building multiple transliteration engines.

In 2010, Antony P.J tried to address the problem of transliterating English to Kannada language using SVM kernel. The transliteration scheme was modeled using sequence labeling problem. The framework was based on data driven approach and one to one mapping method was used to simplify the development of transliteration system. The transliteration module uses an intermediate code, which is designed for preserving the phonetic properties [20]. Ekbal, Naskar and Bandyopadhyay made substantial contribution to develop transliteration systems for Indian languages to English and especially for Bengali-English transliteration [21-28]. Manoj K. Chinnakotla demonstrated a transliteration system for resource scare languages by using statistical methods to monolingual resources in conjunction with hand crafted bilingual rules [29]. The statistical technique used by them was Character Sequence Modeling (CSM). They proved that if the origin of the word is used for the transliteration, then the system performs better than statistical methods. In 2012, Kishorjit Nongmeikapam proposed a Phoneme-based method to transliterate Bengali script to Meitei Mayek script using SVM and was based on Part of Speech (POS) tagging of the Bengali Script text and transliterated to Meitei Mayek after POS tagging. After POS tagging they used the YamCha toolkit to train the corpus. They used polynomial kernel function and the pair wise multi-class decision to obtain the transliteration [30].

3. SUPPORT VECTOR MACHINE (SVM)

SVM does the classification by constructing an n-dimensional hyperplane which optimally segregates the data into two partitions. SVM based models are similar to the neural network models. Theoretically, SVM model with sigmoid kernel function is similar to a two-layer neural network. SVM is a new avatar of kernel functions with a supervised learning approach. It learns from a set of inputs values with the associated output values. It constructs a hyperplane between two classes using binary classifier. Basically SVM is a binary classifier in which data points are classified in two classes with +1 and -1 labels. While separating input examples in two classes it maximise the separation between two classes using the method called as max margin. Due to max margin separation error rate gets minimised and if any new input with unknown label arrives for classification, the chances of making error is minimised.

Let the data set is $\{x_1, x_2, ...x_n\}$ and the desired output or class label is $y_i \in \{+1,-1\}$, then two boundary planes and hyper plane is obtained by using following equations eq(1), eq(2) and eq(3).

$$(w^T x_i - \gamma) >= +1 \text{ ----------eq(1)}$$

$(w^T x_i - \gamma) <= -1$----------eq(2)
$(w^T x_i - \gamma) = 0$ ----------eq(3) where 1<=i<=n.

The data points should satisfy the equation eq(1), eq(2) and eq(3) for correct classification. The decision boundary can be calculated with the following optimization problem

Minimize ½ $|| w||^2$

In few cases application allows misclassifications where small amount of error is tolerated. In such cases, the degree of misclassification can be measured by using the slack variable ξ and C as a control parameter. After introducing slack variable, equations eq(1), and eq(2) can be written as

$(w^T x_i - \gamma) >= +1-\xi$ ---------- eq(4)
$(w^T x_i - \gamma) <= -1+\xi$---------- eq(5)

Now the problem is minimised under the constraint as Minimize ½ $|| w||^2 + C \Sigma \xi_i$
SVM also allows non linear mapping if the data set is not linearly separable in a high dimensional space. In this case, it uses a non-linear kernel function for constructing the new feature space. SVM can be used for multiclass data set where number classes can be k. In case of binary classifier, one dimensional plane is divided in two subspaces while for multiple classes; it divides the hyper plane in multidimensional subspaces [31].

4. DEVANAGARI SCRIPT

Hindi and Marathi languages are written using Devanagari script. Devanagari script used for Hindi and Marathi have 12 pure vowels , 2 loan vowels from the Sanskrit language and 1 loan vowel from English. There are total 34 consonants, 5 conjuncts, 7 loan consonants and 2 traditional signs in Devanagari script and each consonant have 14 variations through integration of 14 vowels [32-34]. Table 1 shows Devanagari script along with their equivalent phonetic mapping in Roman. The consonant /ळ/ is used only in Marathi and not in Hindi.

Table 1: Devanagari Script

Pure consonants					**Vowel**	**Matra**	**Vowel**	**Matra**
क→ka	ख→kha	ग→ga	घ→gha	ङ→nga	अ→a	No Sign	ऋ→RU	ृ
च→cha	छ→chha	ज→ja	झ→jha	ञ→ya	आ→ A	ा	ए→E	े
ट→Ta	ठ→Tha	ड→Da	ढ→dha	ण→Na	इ→i	ि	ऐ→ai	ै
त→ta	थ→tha	द→da	ध→Dha	न→na	ई→ee	ी	ओ→oo	ो
प→pa	फ→pha	ब→ba	भ→bha	म→ma	उ→u	ु	औ→au	ौ
य→ya	र→ra	ल→la	व→va	श→sha	ऊ→U	ू	अं→am	ं/ ँ
ष→Sha	स→sa	ळ→La	ह→ha		ऋ→Ru	ृ	अ: →aH	ः
Conjucts and Symbols →		क्ष→ksha	ज्ञ→dnya	श्र→shra	द्य→dya	त्र→Tra	ॐ→om	श्री-Shri
Loan Letters →		ढ़→Dhxa	ख़→khxa	ग़→gxa	ज़→jxa	फ़→phxa	ड़→Dxa	क़→kxa

5. SYSTEM ARCHITECTURE

The overall system architecture of proposed methodology is depicted in Figure 1. YamCha toolkit is used for the implementation of SVM. This section focuses on the various steps needed to obtain the transliteration of named entity written in Hindi and Marathi using Devanagari script in English using Roman script. The overall logical flow of the transliteration system is divided into following three modules.

- Preprocessing on raw input data
- Training of bilingual corpus using polynomial linear function of degree two
- Testing of additional data

5.1. Preprocessing

Preprocessing phase is used to convert the raw input into system acceptable format. Raw input is the set of NE strings or words which cannot be directly processed by the system. These NEs need to be normalized into the system specified format. 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 is treated as a one phonetic unit. The overall preprocessing is achieved by dividing task into following two sub modules.

- Syllabification
- Alignment using phonetic mapping

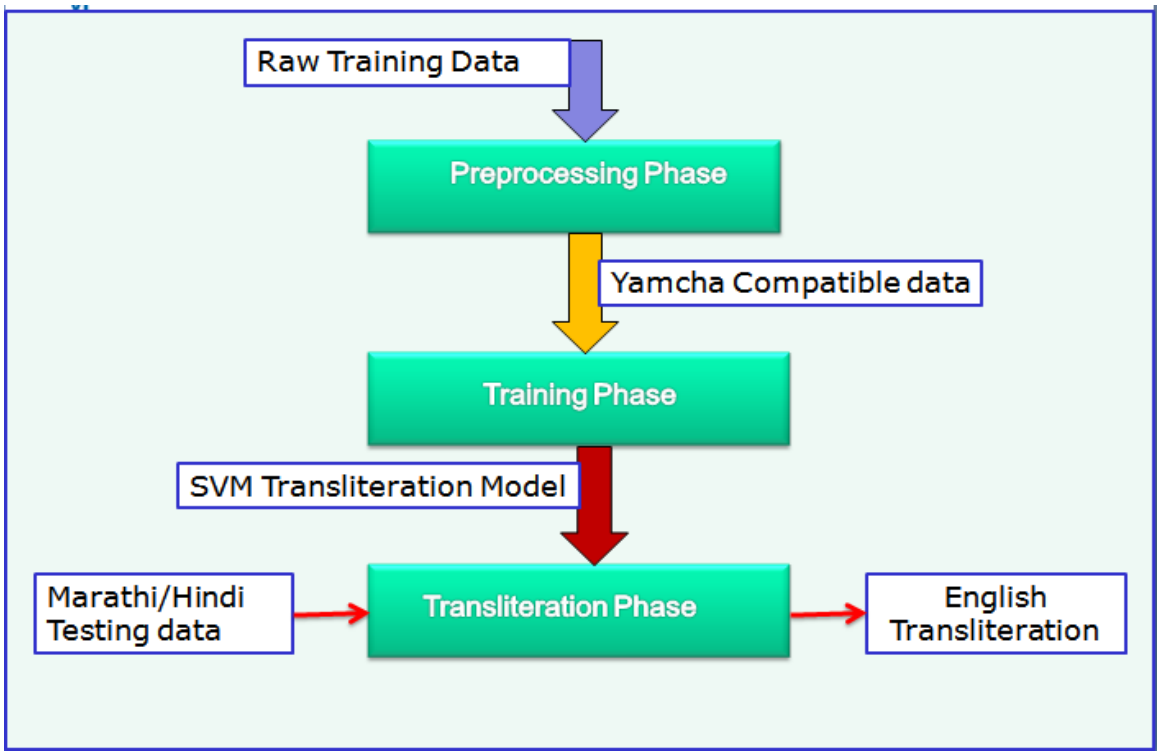


Figure1. System Architecture

5.1.1 Syllabification

Syllabification refers to the segmentation of source and target language NEs into source and target language Translation Units (TU). Translation unit is equivalent to the phonetic unit or syllabic unit of the source or target language. For example, the NE in Hindi and Marathi written using Devanagari 'प्रविण' (person name) is syllabified as प्र + वि + ण whereas its English equivalent 'pravin' is syllabified as pra + vi + n using its phonetic mappings [35]. The YamCha tool accepts the input into syllabic units, which are called as labels. As the each input is equivalent to phonetic unit, phoneme is taken as one of the feature in this implementation. The target language TU is given as a tag to the given input which is a phonetic equivalent of Hindi and Marathi in English. Hindi and Marathi NEs form the source language TUs whereas the corresponding phonetic units in English form target language TUs. The Hindi and Marathi NEs can be syllabified into its corresponding TUs by using following Algorithm 1.

Algorithm 1

Input: W is the NE in Hindi or Marathi to be syllabified.

// Pointer initially points to first Source Transliteration Unit (STU).
foreach W **do** // For every character in word W, perform following steps:
1. Check if the character is a consonant or a vowel.
2. If the character is a consonant then add the character to current STU.
3. If the character is a vowel then
   a. Add the character to current STU.
   b. Increment the pointer so that it points to the next STU.

end foreach

Output: Phonetic Units in Hindi and Marathi

Using Algorithm 1, Hindi and Marathi NEs are syllabified into its phonetic units. Following are the few examples showing how syllabification is done.

Name in Devanagari→ महाराष्ट्र	STUs → [म | हा | रा | ष्ट्र]
Name in Devanagari→ ओंकारेश्वर	STUs → [ओं |का| रे |श्व |र]
Name in Devanagari→ नोवरोझाबाद	STUs → [नो | व | रो | झा | बा | द]
Name in Devanagari→ अब्दुल्लाहगंज	STUs → [अ | ब्दु | ल्ला | ह | गं | ज]
Name in Devanagari→ निरंजनकुमार	STUs → [नि | रं | ज | न | कु | मा | र]
Name in Devanagari→ नारायणगावकर	STUs → [ना | रा | य | ण | गा | व | क | र]
Name in Devanagari→ त्रिभुवननारायण	STUs → [त्रि | भु | व | न | ना | रा | य | ण]

5.1.2 Alignment using phonetic mapping

The raw Hindi or Marathi to English mapping from previous step is of the form मा → ma, णि → ni, क → k, etc. But, in certain examples like 'वसंतराव' (व सं त रा व), which is syllabified as (va

san t ra v), a consonant /व/ is mapped to both ‘va’ as well as ‘v’. It is difficult to automate such possibilities. In fact, these are the rules that are going to get trained using tool. Such possibilities were edited manually to get proper bilingual corpus for training. Then each STU is mapped to a Target Language Transliteration Unit (TTU). Following are the few examples showing how syllabification and phonetic mapping is done.

NE in Devanagari		**STUs**	**TTUs**
महाराष्ट्र	→	[म | हा | रा | ष्ट्र]	→ [ma | ha| ra| shtra]
ओंकारेश्वर	→	[ओं |का| रे |श्व |र]	→ [om | ka| re| shwa | r]
नोवरोझाबाद	→	[नो | व | रो | झा | बा | द]	→ [no | v | ro | jha| ba| d]
अब्दुल्लाहगंज	→	[अ | ब्दु | ल्ला | ह | गं | ज]	→ [a | badu| lla | h| gan | j]
निरंजनकुमार	→	[नि | रं | ज | न | कु | मा | र]	→ [ni | ran | ja | n | ku | ma | r]
नारायणगावकर	→	[ना | रा | य | ण | गा | व | क | र]	→ [na| ra| ya| n | ga |v | ka | r]
त्रिभुवननारायण	→	[त्रि | भु | व | न | ना | रा | य | ण]	→ [tri | bhu | va | n | na | ra | ya | n]

5.2. Training Phase

Training phase requires two things, one is training data and other is feature on which the data is to be trained. Parallel data obtained during syllabification is arranged in the YamCha required format and then n-gram features are used to train this data. The first step in using the YamCha is to create training files. In this file the first column represents a Hindi and Marathi syllabified named entity and second column represent a true tag or label corresponding to a Hindi and Marathi syllabified unit in English. In this phase the classification is done based on the n-grams. The details of classification are given for four different n-grams.

- Bigrams
- Trigrams
- Fourgrams
- Fivegrams

5.2.1. Bigram

The bigram is the first feature used to train the parallel corpora. The bigram feature creates the SVM classes by considering current and immediate next STU. The creation of classes is described using two NEs ‘प्रविण’ and ‘माणिकराव’ written in Devanagari script. After training the above dataset, SVM kernel created eight different patterns in total using the bigram feature. Using bigram feature the patterns for NE ‘प्रविण’ would be {प्रवि, विण ,ण} and for NE ‘माणिकराव’ the patterns would be{ माणि, णिक, करा, राव, व }. Classifications of these eight patterns are shown in the Figure 2. Few patterns are classified into negative space and others are in positive space by the SVM kernel function.

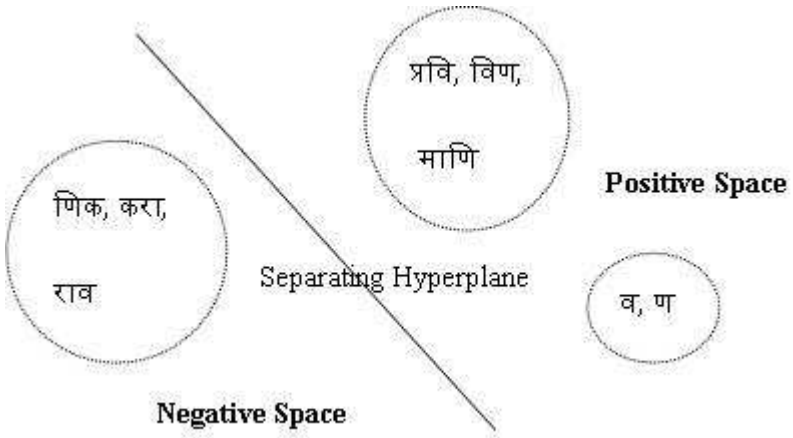


Figure2. Hyperplane for Bigram

5.2.2. Trigram

The trigram is the second feature used to train the parallel corpora. The trigram feature creates the SVM classes by considering current and immediate next two STUs. Using trigram feature the patterns for NE 'प्रविण' would be { प्रविण, विण ,ण} and for NE 'माणिकराव' the patterns would be{ माणिक, णिकरा, कराव, राव, व }. Figure 3 shows the vector space of classification.

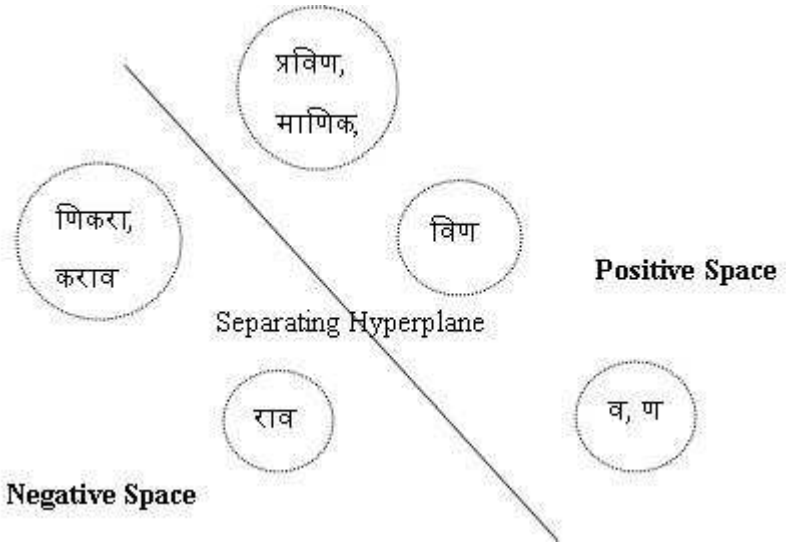


Figure3. Hyperplane for Trigram

5.2.3. Fourgram

The fourgram is the third feature used to train the parallel corpora where SVM classes created by considering current and immediate next three STUs as shown in Figure 4. Using fourgram feature the patterns for NE 'प्रविण' would be { प्रविण, विण ,ण} and for NE 'माणिकराव' the patterns would be{ माणिकरा , णिकराव , कराव, राव, व }.

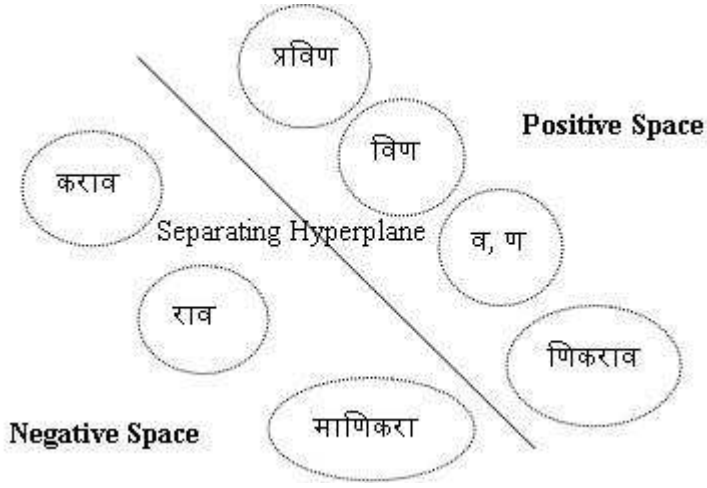


Figure4. Hyperplane for Fourgram

5.2.4. Fivegram

The fivegram is the fourth feature used to train the parallel corpora where SVM classes created by considering current and immediate next four STUs as shown in Figure 5. Using fivegram feature the patterns for NE 'प्रविण' would be { प्रविण, विण ,ण} and for NE 'माणिकराव' the patterns would be{ माणिकराव , णिकराव , कराव, राव, व }.

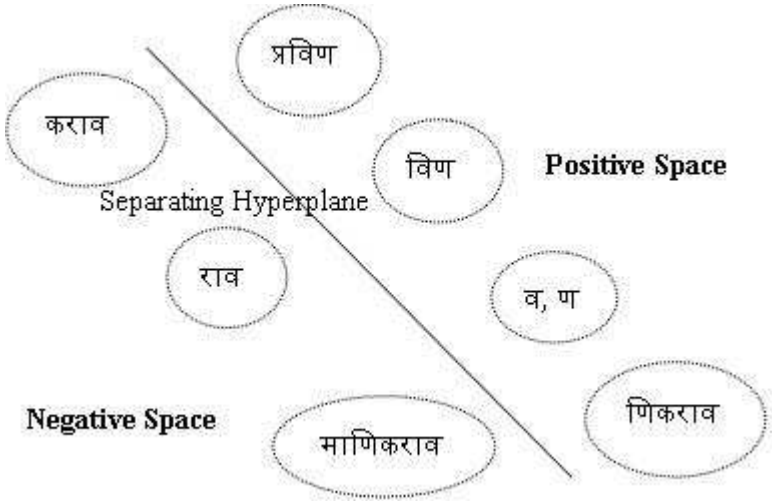


Figure6. Hyperplane for Fivegram

5.3. 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 hyperplane. 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 figures 2 to 5, then generates the correct output. If the pattern is not found then it shows garbage output i.e. incorrect English

transliterated output. For example, given an input as "प्रविणराव" to the model file, it produces output as

प्र →pra

वि→vi

ण→n

रा→ra

व→v

6. EXPERIMENTATION DETAILS

This section describes the overall implementation details of Devanagari to Roman machine transliteration using YamCha tool.

6.1. Configuration

The programming implementation is carried out using java and eclipse integrated development environment. The GUI is designed in Netbeans because it supports the inbuilt control component i.e. Button, Textview etc. After the training, model file is generated. The model file is executed in Windows platform using YamCha.exe and libyamcha.dll files. The training file is stored with dot data extension, UTF-8 and Linux end line. The training is carried out on Linux platform.

Features selected are variable sized n-grams i.e. Bigram, Trigram, Fourgram and Fivegram as a window size. N-grams are generated using only forward movement. The Makefile is a file where the feature parameters are modified according to n-gram requirement. The following summary gives details of each n-gram,

For window size of two (Bigram)

```
SVM_PARAM = -t 1 -d 2 -c 1
FEATURE   = F:0..1:0..0
MULTI_CLASS = 2
```

For window size of three (Trigram)

```
SVM_PARAM = -t 1 -d 2 -c 1
FEATURE    = F:-1..1:0..0
MULTI_CLASS = 2
```

For window size of four (Fourgram)

```
SVM_PARAM = -t 1 -d 2 -c 1
FEATURE    = F:-1..2:0..0
MULTI_CLASS = 2
```

For window size of five (Fivegram)

```
SVM_PARAM = -t 1 -d 2 -c 1
FEATURE   = F:-2..2:0..0
MULTI_CLASS = 2
```

The SVM_PARAM "-t 1 -d 2 -c 1" means that second degree of polynomial kernel and one slack variable is used. The YamCha tool supports only polynomial kernel. The MULTI_CLASS=2 means one versus rest i.e. the score indicates the distance from separating hyperplane.

For the training two parameters are mandatory. First one is the location of file which is a bilingual corpus written in the training format and second is the prefix name of the model file.
The parsing direction is from Left to Right for Devanagari script. We have chosen the default setting which is forward parsing mode (Left to Right). The Makefile is used for setting different parameters for bigram, trigram, fourgram and fivegram.

6.2. Feature Selection

The parameter FEATURE is used to change the feature sets (window size) for dataset. The default setting is "F:-2..+2:0.. T:-2..-1". F denotes the static feature while T denotes the dynamic feature. The static feature "F:-2..+2:0.." means that [beginning positing of token] is -2 and [end position of token] is +2.which is shown in Figure 6.

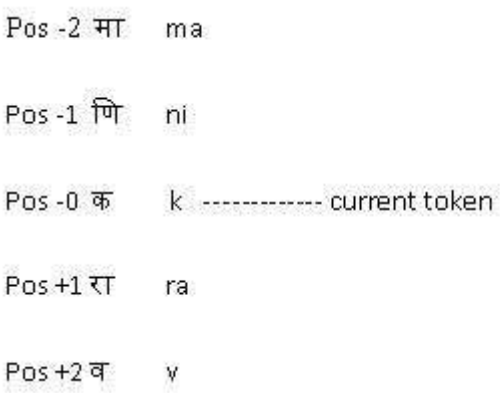


Figure6. Feature Selection for Window Size -2 to +2

6.3. Testing and Results

The system was tested for person names, historical place name, city names of Indian origin. Standard bilingual corpus in Unicode format for Hindi and Marathi is not available; hence the test data set is created from voters' lists of State of Maharashtra, census website of Government of India, road atlas of various states etc. Initially, 10k NE data set is trained. Figure 7 shows the snapshot of machine transliteration.

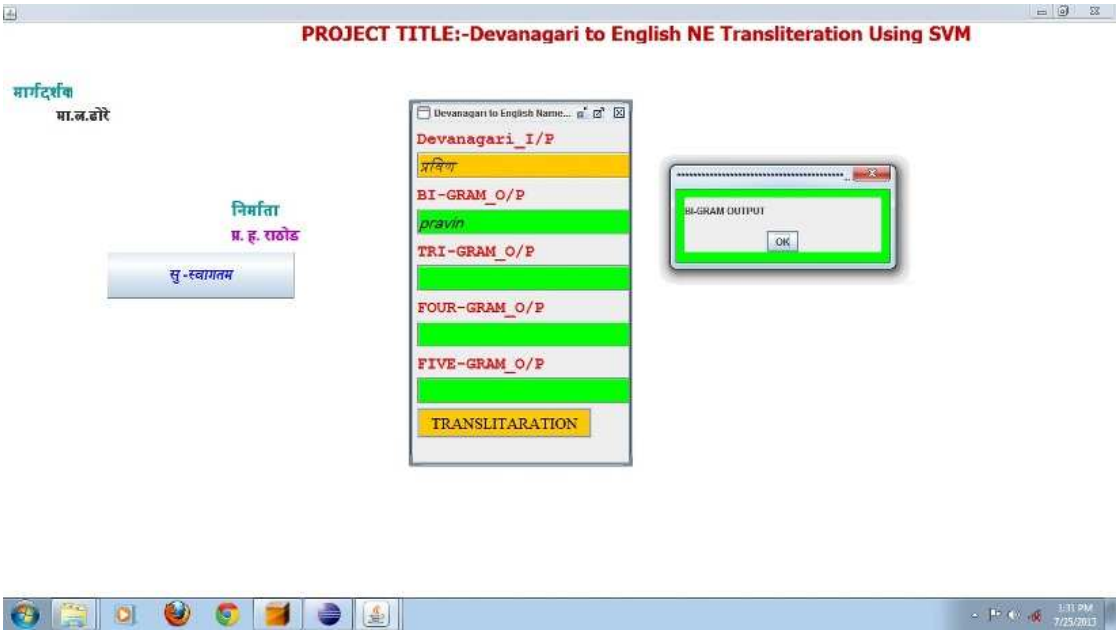


Figure7. Snapshot of Transliteration Tool

Accuracy (ACC) which is also known as Word Error Rate (WAR) measures the correctness of the transliteration candidate produced by a transliteration system. ACC = 1 means that candidate is the correct Transliteration i.e. it matches with the reference, and ACC = 0 means candidate is not correct. Accuracy is calculated by using the following formula.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1, if\ correct\ match\ found \\ 0, if\ incorrect\ match\ found \end{cases}$$

where N is total number of NEs. Initially, over the trained data set of 10k size, test data set of 5k size is tested and we obtained the results depicted in Table 2 and Figure 8.

Table2. Initial Results

	Bi-gram	**Tri-gram**	**Four-gram**	**Five-gram**
Number of NEs Trained	10000	10000	10000	10000
Number of NEs Tested	5000	5000	5000	5000
Number of Correct NEs	2098	2266	2520	1696
Accuracy	41.96%	45.32%	51.96%	33.92%

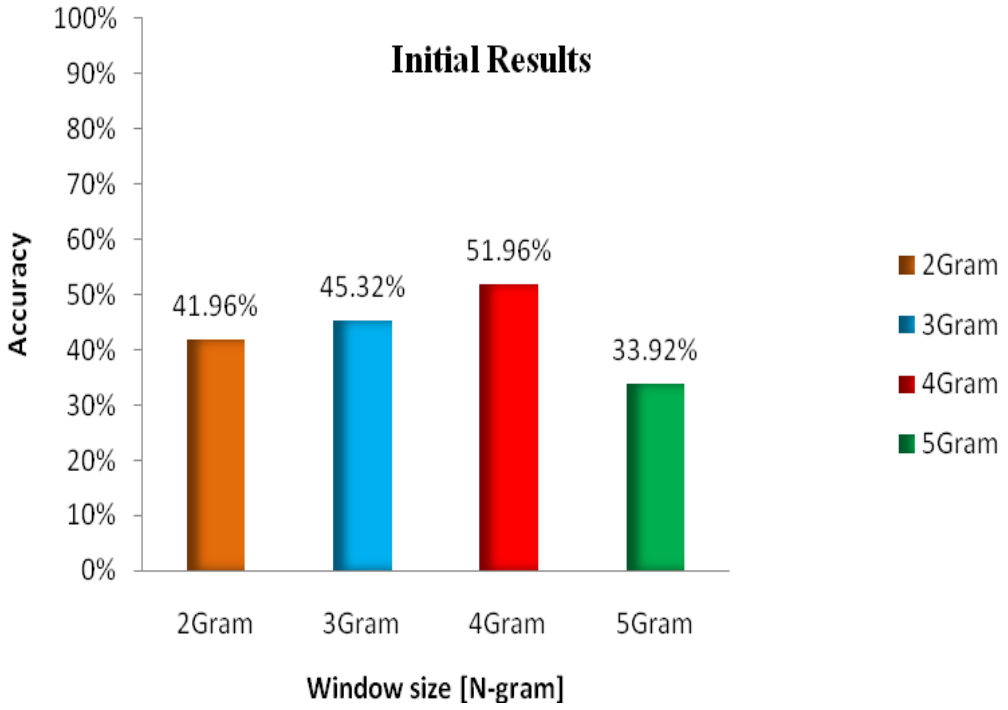


Figure 8. Initial Results

The initial results of the transliteration were very disappointing as compared to the results obtained by other researches for Tamil and Kannada languages. After in depth analysis of the trained data, it has observed that results were less as the pattern created was not having location names ending with 'wadi', 'gaon' , 'nagar' etc as well as the surnames ending with 'kar' , 'pure', 'kare' etc. After adding 500 commonly used suffixes in Hindi and Marathi, in the training data set, we achieved reasonable transliteration accuracy as shown in Table 3 and Figure 9.

Table3. Final Results

	Bi-gram	**Tri-gram**	**Four-gram**	**Five-gram**
Number of NEs Trained	10500	10500	10500	10500
Number of NEs Tested	5000	5000	5000	5000
Number of Correct NEs	3387	3824	4307	4327
Accuracy	67.74%	76.48%	86.14%	86.52%

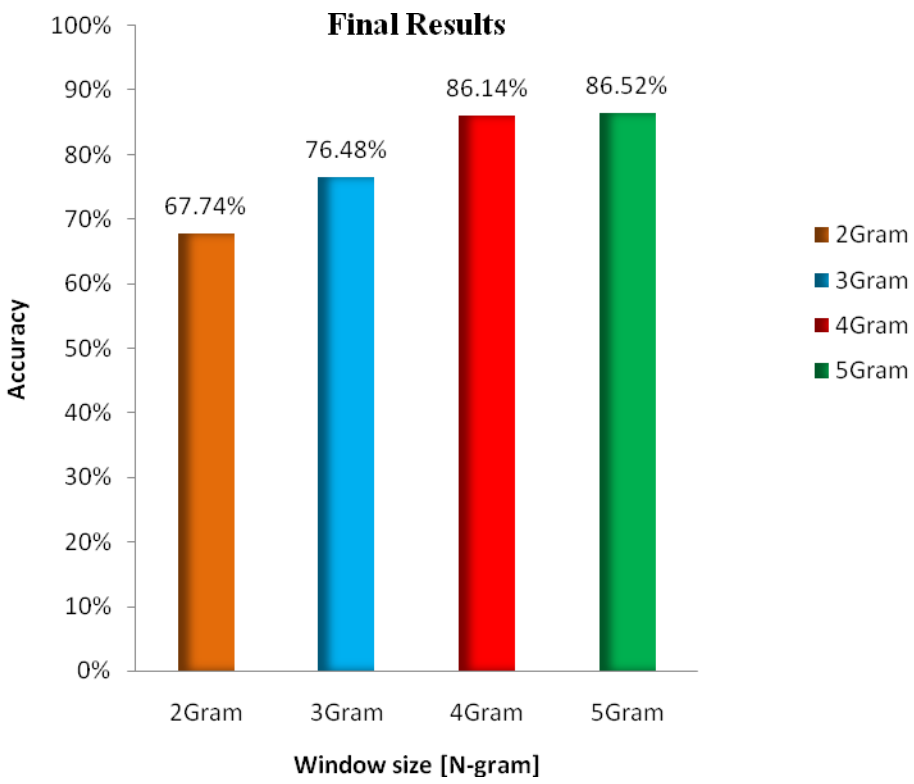


Figure 9. Final Results

Although we have not found any other specific work for the Hindi-English and Marathi-English language pairs using SVM, our results are compared to the other language pairs only for the languages in India. Table 4 depicts the comparison of performance with the other different language pairs in India using SVM.

Table4. Comparison of Results

Language Pair	**Classification Approach**	**Top-1 Accuracy Performance (%)**	**Metric**
English - Malayalam	SVM	90.00	Word Accuracy
English - Kannada	SVM	81.25	Word Accuracy
Bengali - Meitei Mayek	SVM	86.04	Word Accuracy
English - Tamil	SVM	84.16	Word Accuracy
Hindi/Marathi- English	SVM	86.52	Word Accuracy

7. CONCLUSIONS

In this paper, we presented machine transliteration for Hindi to English and Marathi to English language pairs using Support Vector Machine (SVM). As the Hindi and Marathi are phonetically rich languages, phoneme is selected as a prime feature. As the Devanagari NEs written in Hindi and Marathi consists of the segments of two, three, four and five, a variable sized n-gram is taken as second feature. We used SVM as a machine learning algorithm for the classifications of patterns based on phoneme and variable n-gram sizes. In our case, transliteration is treated as sequence labeling problem in which output of current label depends on more than one previous and future label. It is desirable that transliteration model takes care of all the dependencies. As SVM creates the hyperplane using linear polynomial function, there is no restriction on number of classes to be generated. As SVM can generate adequate number of classes for all available patterns, it is more suitable for the transliteration task. In sequence labeling, it has been observed that increase in n-gram size i.e. from bigrams to fivegrams improves the accuracy. As there are no NEs consisting of one syllabic unit, it has been observed that unigram feature provides very less accuracy. Accuracy has been gradually increased as the n-gram sized increased. The bigram gives good accuracy only to the NEs having length two. Similarly, trigram gives good results for the NEs having length three. The fourgram and fivegram accuracy is very close. It has also been observed that sixgram results are almost same as fivegrams but unnecessary increases training time compared to fivegram. We have trained 10.5 k corpus using YamCha Toolkit and tested for 5k data. The top-1 accuracy obtained for 5k testing data using bigram is 67.74%, using trigram is 76.48%, using fourgram is 86.14% and using fivegram is 86.52%. We conclude that, 5-gram is best suitable size for Hindi and Marathi to English named entity transliteration. The current system is tested for person names and place names only. It can further be extended for foreign names, organization names. As English is non phonetic language, we have not carried out back transliteration. Therefore, there is also future scope to perform the back transliteration.

REFERENCES

[1] Padariya Nilesh, Chinnakotla Manoj, Nagesh Ajay, Damani Om P.(2008) “Evaluation of Hindi to English, Marathi to English and English to Hindi”, IIT Mumbai CLIR at FIRE.

[2] Saha Sujan Kumar, Ghosh P. S, Sarkar Sudeshna and Mitra Pabitra (2008) “Named entity recognition in Hindi using maximum entropy and transliteration.”

[3] BIS (1991) “Indian standard code for information interchange (ISCII)”, Bureau of Indian Standards, New Delhi.

[4] Joshi R K, Shroff Keyur and Mudur S P (2003) “A Phonemic code based scheme for effective processing of Indian languages”, National Centre for Software Technology, Mumbai, 23rd Internationalization and Unicode Conference, Prague, Czech Republic, pp 1-17.

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

[6] Knight Kevin and Graehl Jonathan (1997) “Machine transliteration”, In proceedings of the 35th annual meetings of the Association for Computational Linguistics, pp 128-135.

[7] Stalls Bonnie Glover and Kevin Knight (1998) “Translating names and technical terms in Arabic text.”

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

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

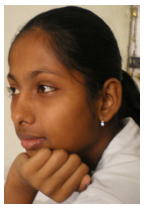
[10] Jung S. Y., Hong S., S., Paek E.(2003) "English to Korean transliteration model of extended Markov window", In Proceedings of the 18th Conference on Computational Linguistics, pp 383–389.
[11] Ganapathiraju M., Balakrishnan M., Balakrishnan N., Reddy R. (2005) "OM: One Tool for Many (Indian) Languages", ICUDL: International Conference on Universal Digital Library, Hangzhou.
[12] Malik M G A (2006) "Punjabi Machine Transliteration", Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL, pp 1137–1144.
[13] Sproat R.(2002) "Brahmi scripts, In Constraints on Spelling Changes", Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands.
[14] Sproat R.(2003) "A formal computational analysis of Indic scripts", In International Symposium on Indic Scripts: Past and Future, Tokyo.
[15] Sproat R.(2004) "A computational theory of writing systems, In Constraints on Spelling Changes", Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands.
[16] Kopytonenko M. , Lyytinen K. , and Krkkinen T.(2006) "Comparison of phonological representations for the grapheme-to-phoneme mapping, In Constraints on Spelling Changes", Fifth International Workshop on Writing Systems, Nijmegen, The Netherlands.
[17] Ganesh S, Harsha S, Pingali P, and Verma V (2008) "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.
[18] Sumaja Sasidharan, Loganathan R, and Soman K P (2009) "English to Malayalam Transliteration Using Sequence Labeling Approach" International Journal of Recent Trends in Engineering, Vol. 1, No. 2, pp 170-172
[19] Oh Jong-Hoon, Kiyotaka Uchimoto, and Kentaro Torisawa (2009) "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
[20] Antony P.J, Soman K.P (2010) "Kernel Method for English to Kannada Transliteration", Conference on Machine Learning and Cybernetics, pp 11-14
[21] Ekbal A. and Bandyopadhyay S. (2007) "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.
[22] Ekbal A. and Bandyopadhyay S. (2008) "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.
[23] Ekbal A. and Bandyopadhyay S. (2008), "Development of Bengali named entity tagged corpus and its use in NER system", In Proceedings of the 6th Workshop on Asian Language Resources.
[24] Ekbal A. and Bandyopadhyay S. (2008) "A web-based Bengali news corpus for named entity recognition", Language Resources & Evaluation, vol. 42, pp 173–182.
[25] Ekbal A. and Bandyopadhyay S.(2008) "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.
[26] Ekbal A. and Bandyopadhyay S.(2009) "Bengali Named Entity Recognition using Classifier Combination", In Proceedings of Seventh International Conference on Advances in Pattern Recognition, pp 259–262.
[27] Ekbal A. and Bandyopadhyay S. (2009) "Voted NER system using appropriate unlabelled data", In Proceedings of the Named Entities Workshop, ACL-IJCNLP.
[28] Ekbal A. and Bandyopadhyay S. (2010) " Named entity recognition using appropriate unlabeled data, post-processing and voting", In Informatica, Vol 34, No. 1, pp 55-76.
[29] Chinnakotla Manoj K., Damani Om P., and Satoskar Avijit (2010) "Transliteration for Resource-Scarce Languages", ACM Trans. Asian Lang. Inform,Article 14, pp 1-30.
[30] Kishorjit Nongmeikapam (2012) "Transliterated SVM Based Manipuri POS Tagging", Advances in Computer Science and Engineering and Applications, pp 989-999
[31] K.P.Sonam, V. Ajay, R. Laganatha.(2009) "Machine Learning with SVM and Other Kernel Methods", Machine Learning Book, PHI.
[32] Koul Omkar N. (2008) "Modern Hindi Grammar", Dunwoody Press
[33] Walambe M. R. (1990) "Marathi Shuddalekhan", Nitin Prakashan, Pune
[34] Walambe M. R. (1990) "Marathi Vyakran", Nitin Prakashan, Pune

[35] Dhore M L, Dixit S K and Dhore R M (2012) "Hindi and Marathi to English NE Transliteration Tool using Phonology and Stress Analysis", 24th International Conference on Computational Linguistic,s Proceedings of COLING Demonstration Papers, at IIT Bombay, pp 111-118

Authors

P. H. Rathod (pravin.rathod@vit.edu) has completed BE in Information Technology, from Government College of Engineering, Karad, Maharashtra, India, in 2008. Recently he has completed ME in Computer Science and Engineering from Vishwakarma Institute of Technology, Pune, India in 2013. Currently he is working as Assistant Professor in Department of Computer Engineering at Vishwakarma Institute of Technology, Pune. He has his interest in Machine Translation and Machine Transliteration specifically in Devanagari-English Language Pairs. His current areas of research are Mobile Ad hoc Networks, Internet Routing Algorithms, Computer Networking, Machine Translation and Transliteration

M. L. Dhore (manikrao.dhore@vit.edu) has completed ME in Computer Science and Engineering from NITTR, Chandigarh, India in 1998. Currently he is working as Associate Professor in Department of Computer Engineering at Vishwakarma Institute of Technology, Pune. Presently he is pursuing his Ph.D. from University of Solapur, Maharashtra, India, in the area of Computational Lingui stics. He has his interest in Machine Translation and Machine Transliteration specifically in Marathi-English and Hindi- English Language Pairs. He has developed the tools for Devanagari to English Machine Transliteration for the online web based commercial applications. His current areas of research are Internet Routing Algorithms, Computer Networking, Machine Translation and Transliteration.



Ruchi M Dhore (ruchidhore93@gmail.com) is the student of Third Year Computer Engineering at Pune Vidyarthi Griha's College of Engineering and Technology, Pune, Maharashtra, India. She is scholar student of her college and securing distinction every year in the University of Pune examinations. She is very good in programming and won the prizes in state level and national level competitions. Her area of research interest includes Text Processing and Pattern Searching. She likes to build her carrier in the development of language processing tools for Marathi language.