Handling of Non-Finite Verbs in English to Sanskrit Machine Translation

Vimal Mishra[*]
vimal.mishra.upte@gmail.com

Abstract

In this paper, we discuss non-finite verbs viz. gerunds, infinitives and participles in our English to Sanskrit machine translation system. Our system is an integrated model of a rule based approach of machine translation with Artificial Neural Network (ANN) model that translates an English sentence into Sanskrit sentence. We use feed forward ANN for the selection of Sanskrit word like noun, verb, object, adjective etc from English to Sanskrit User Data Vector (UDV). Our system uses only morphological markings to identify subject, object, verb, preposition, adjective, adverb, conjunctive and as well as non-finite verbs viz. gerunds, infinitives and participles type of sentences. We propose rules for handling of non-finite verbs in our EST system. The performance of our system is encouraging.

1. Introduction

India is a multilingual country with eighteen constitutionally recognized languages (Sinha & Jain, 2003). Sanskrit language is the mother of all Indian languages. Mostly, all Indian languages are originating from Sanskrit language. There have been many MT systems for English to other foreign languages as well as to Indian languages but few for English to Sanskrit machine translation. So, if we develop English to Sanskrit machine translation system then it will easier to translate from English to any Sanskrit originating Indian languages. Some works on Sanskrit parser and morphological analyzers have done earlier which is briefly described below.

The work of Ramanujan (1992) describes that morphological analysis of Sanskrit is the basic requirement for the computer processing of Sanskrit. The Nyaya (Logic), Vyakarana (Grammar) and Mimamsa (Vedic interpretation) are suitable solutions that

[*] Head, Department of Computer Engineering, Government Girls Polytechnic, Allahabad-211004. U.P., India.

cover syntactic, semantic and contextual analysis of Sanskrit sentence. P. Ramanujan has developed a Sanskrit parser 'DESIKA', which is Paninian grammar based analysis program.

Briggs (1985) uses semantic nets (knowledge representation scheme) to analyze sentences unambiguously. He compares the similarity between English to Sanskrit and provides the theoretical implications of their equivalence.

Huet (2003) has developed a Grammatical Analyzer System, which tags NPs (Noun Phrases) by analyzing sandhi, samasa and sup affixation.

The works in Sanskrit processing tools and Sanskrit authoring system have carried out Jawaharlal Nehru University, New Delhi-India. It is currently engaged in karaka Analyzer, sandhi splitter and analyzer, verb analyzer, NP gender agreement, POS tagging of Sanskrit, online Multilingual amarakosa, online Mahabharata indexing and a model of Sanskrit Analysis System (SAS)(Jha et.al ,2006).

Mishra & Mishra (2008) have presented study of example based English to Sanskrit machine translation that shows example based machine translation (EBMT) has emerged as one of the most versatile, computationally simple and accurate approaches for English to Sanskrit machine translation in comparison to rule based machine translation (RBMT) and statistical based machine translation (SBMT). They show with illustrative examples the divergence between Sanskrit and English languages which can be considered as representing the divergences between the order free and SVO (Subject-Verb- Object) classes of languages.

Hossain (2008) has worked on "Anubadok" system that translates English sentences into Bengali sentences. Basically, the Bengali language follows S-O-V structure that is most similar to Sanskrit language. Anubadok is rule based machine translation system that uses Penn TreeBank annotation system for part-of-speech tagging.

Kadambini, K. et. al (2009) have presented an algorithm that translates English texts to Sanskrit based on the link grammar formalism for parsing natural language texts to Sanskrit Language. They have formed linkages between two words in the input sentence that are modified into karaka relations. Their proposed algorithm converts the given English texts into a sequence of sentences. These sentences are passed to the Link Parser whose output specifies the relations that exist between the words in the form of links. They claim that the links are appropriately converted into the corresponding karaka roles. After translating these nouns and verbs, they are added the suffixes to the words by noun and verb generator programs.

Goyal & Sinha (2009) have presented a study towards design of English to Sanskrit machine translation system. Their work is demonstration of machine translation of English to Sanskrit for simple sentences based on PLIL (Pseudo Lingua Intermediate Language) generated by AnglaBharti and Astadhyayi (Panini grammar) rules. They claim that their system translates affirmative, negative, interrogative, imperative, active and passive voice sentences.

We have developed a prototype model of English to Sanskrit machine translation (EST) system using ANN model and rule based approach. ANN model gives matching of equivalent Sanskrit word of English word which handles noun and verb. The ANN based system gives us faster matching of English noun (subject or object) or verb to appropriate Sanskrit noun (subject or object) or dhaatu. The rule based model generates Sanskrit translation of the given input English sentence using rules that generate verb and noun for Sanskrit. The rule based approaches mostly make use of hand written transfer rules to the translation of substructures from source language (English sentence) to target language (Sanskrit sentence). The main advantages of rule based approaches are easy implementation and small memory requirement (Jain et. al, 2001).

We have divided our work into the following sections. Section 2 presents non-finite verbs viz. gerunds, infinitives and participles in English and Sanskrit that describe the rules for forming words of non-finite verbs in Sanskrit which are based on Panini grammar. Section 3 describes the system model of our EST system. Section 4 presents implementation and the illustration with examples as well as the result of the translation in GUI form. The evaluations are shown in section 5. The conclusions are mentioned in section 6.

2. Non-Finite Verbs in English and Sanskrit

In English, Non-finite verbs are gerunds, infinitives and participles. A Gerund is that form of verb which ends in –ing, and has a force of a noun and verb (Wren and Martin, 1994). In English, the gerund acts as a noun in a sentence. It can be used as a subject or object in sentence. There are different types of gerund used in the English sentences which are given below.

In English, the gerund acts as a subject of a verb in a sentence. The example is given below.

1. Working is useful.
2. Sleeping is necessary for life.
3. Playing cricket makes us active.

In English, the gerund acts as an object of a verb in a sentence. The example is given below.

4. Stop playing.
5. I like reading books.
6. He enjoyed sleeping in open air.

In English, the gerund is governed by a preposition in a sentence. The example is given below.

7. He is tiered of waiting.
8. I am fond of eating mangoes.
9. You will be punished for stealing a pen.

In English, when gerund is used to combine two sentences, the preposition comes before gerund. The example is given below.

10. After finishing his work, the boy went to school.
11. On seeing his friend, he was very pleased.

12. Having gone to town, Ram drinks water.

In Sanskrit, the gerund is handled by ktvaa (or tvaa), lyut (an) and lyap affixes (Egenes, 2000). In Panini grammar, the rule for gerund is given below.

Rule 1: Samankrtrikayah Purvakaale |3|4|2|| (Nautiyal, 1997)

In ktvaa (or tvaa), the suffix used for forming the gerund from simple verb is tvaa. The word formation using ktvaa (or tvaa) affix is given below in table I.

Table I: Word Formation using ktvaa (or tvaa) Affix

S. No.	English gerund	Sanskrit gerund
1	Having done	krtvaa (from krit dhaatu)
2	Having spoken	Uktvaa (from ukta dhaatu)
3	Having gone	Gatvaa (from gata dhaatu)

The situation exits where one verb (action) end then second verb (action) proceeds. There is one necessary condition that both verbs (action) must have same noun (kartaa). If any dhaatu have last character as 'm'or 'n' then eliminate 'm' or 'n' from dhaatu and add tva. According to Panini, the word formation from dhaatu is given below.

Hatva = han+tva
Gatva = gan+tva
Natva = nan+tva

If prefix exits before dhaatu then here present lyap affix, not tva affix. Tva does not change into lyap if there is natr.

Rules for Forming the Lyap Affix

Rule 2: If last character of dhaatu is 'aa', or 'ii', or 'uu', then add 'yaa' as suffix to dhaatu.
/* the following examples shows this rule1 which is given below.
Ut + sthva = utthaayaa (standing)
Aa + nii = aaniiyaa (bringing)
Anu + buu = anubuuyaa (feeling) */

Rule 3: If last cgaracter of dhaatu is 'a', or 'i', or 'u', or 'ri' then add 'tyaa' as suffix to dhaatu.
/* the following examples shows this rule2 which is given below.
Aa + gam = Aagatyaa (coming)
Pra + stu = Prastutyaa (presenting) */

Rule 4: If last character of dhaatu is 'ri.' then add 'iryaa' as suffix to dhaatu.
/* the following examples shows the formation of word using rule3 which is given below.
Vi + kri = vikciryaa (scattering) */

In English, the infinitive is made by adding "to" before verb. The Infinitive works as noun, adjective or adverb in a sentence. There are different types of Infinitive used in the English sentences which are given below.

To work is useful.
I have no time to play.
He comes here to read a book.

In example 13, to work is used as a subject. In example 14, to play is used as an adjective while in example 15, to read is used as adverb. In English, the Infinitive has the following structure.

To +verb (first form)

In Sanskrit, the infinitive is formed, with exception, by:

guna of root + tum (or itum)
The formation of the infinitive is mostly the same as the periphrastic future, only with the krt ending tum, instead of ta. In Sanskrit, the infinitive is handled by tum (itum) affixes (Egenes, 2000).
The Sanskrit infinitive is an indeclinable participle. The infinitive is non-finite form of the verb which is used independently in a sentence. The number and person of the subject in a sentence do not affect on them and it does not indicate any tense of the sentence. In Panini grammar, the rule for infinitive is given below.

Rule 5: Tumanrvulau Kriyaayaam Kriyaarthaayaam |3|3|10| (Nautiyal, 1997)
Rule 6: Kaalsamayavelaasu Tuman |3|3|167| (Nautiyal, 1997)

The infinitive is used as an object (verb) of a verb in English sentence. The infinitive is used as an accusative with the verb that is given in the following examples.

16. ES: He goes to the school to study.
SS: Sah pathitum viddhayaalayam gachchati.
17. ES: Ram goes to see to Shyam.
SS: Raamah Shyamam drashtum yaati.
18. ES: Ram wants to go.
SS: Raamam gantum ichchati.
19. ES: Ram wants to come from the forest.
SS: Raamam vanaad aagantum ichchati.

The infinitive can be used with a passive construction of English sentence that is given in the following examples.

20. ES: The book can be read by the boy.
    SS: Baalen pustakam pathitum shakyate.
21. ES: It is not fit to be heard.
    SS: Shrotum na yajyate.

In the example 16, the English sentence has two verbs such as "goes" and "study". The action of study (that is as a verb) depends on the first verb "goes". So, the second verb is formed with "tuman" affix in Sanskrit translation. The formation of the infinitive word in Sanskrit for some of the commonly used infinitives in English is given in table II.

Table II: Infinitives in Sanskrit and English

Root Dhaatu	**Present**	**Infinitive in Sanskrit**	**Infinitive in English**
ad	atti	attum	to eat
aap	aapnoti	aaptum	to obtain
aas	aaste	aastium	to sit
kr	karoti	kartum	to do
gam	gachchati	gantum	to go

In English, the participles are formed from verbs and acts as an adjectives or verbs in a sentence. There are three types of participles in English such as present participles, past participles and future participle. In English, present participles are usually formed by adding "-ing" to a verb. For example, "glowing" and "being" are present participles. Past participles are usually formed by adding "-ed" or "-en" to a verb. For example, "satisfied" and "spoken" are past participles.

In Sanskrit, there are many types of participles (called kradanta by Panini) such as present active, present middle, present passive, future active, future middle, future passive (gerundive), past active, past passive, perfect active, perfect middle, gerund and infinitive.

In Sanskrit, the participles take krt endings such as primary nominal endings (Egenes, 2000). In Panini grammar, the rule for past passive particle (PPP) is given below.

Rule 7: Kridtind |3|1|93|| (Nautiyal, 1997)

In ta (or ita or na), the suffix used for forming the past passive particle from simple verb is ta. The word formation using ta (or ita or na) affix is given below in table III.

Table III: Word Formation using ktvaa (or tvaa) Affix

S. No.	Root	Present	PPP	English
1	Is	icchati	ishta	desired
2	Gam	gacchati	gata	gone
3	Ji	jayati	jita	conquered
4	Cint	cintayati	cintita	thought
5	tud	tudati	tunna	pushed

3. System Model of Our EST System

We have developed English to Sanskrit machine translation (EST) model that comprised the combination of two approaches: rule based model and the dictionary matching by ANN model. Our EST model has been implemented on window platform using Java. The ANN model is implemented using MATLAB 7.1 neural network tool. We use feed forward ANN that gives matching of equivalent Sanskrit word of English word which handles noun and verb. The rule based model is generated Sanskrit translation of the given input English sentence using rules that extract verb and noun form for Sanskrit. In this paper, we show that the handing of non-finite verbs in our EST model is well. Figure 1 shows the information flow in our EST model.

3.1 Sentence Tokenizer Module

The sentence tokenizer module split the English sentences into tokens (words) using split method of string tokenizer class in Java. The outputs of the sentence tokenizer module are given to POS Tagger module.

3.2 POS Tagger Module

The POS (Part–of-Speech Tagging) is the process of assigning a part–of-speech (such as a noun, verb, pronoun, preposition, adverb and adjective) to each word in a sentence. In POS Tagger module, the Part–of-Speech (POS) tagging is done on each word in the input English sentence. This part–of-speech tagging method falls under the rule-based (linguistic) category. The rule-based taggers use hand coded rules to assign tags to words. The output of POS tagger module is given to rule base engine.

3.3 GNP Detection Module

The GNP detection module detects the gender, number and person of the noun in English sentence. The English language has three genders: masculine, feminine and neuter; two numbers: singular and plural and three persons: first, second and third.

3.4 3.4 Tense and Sentence Detection Module

The English has three tenses: present, past and future; and four forms of each tense such as indefinite, continuous, perfect and perfect continuous. The tense of English sentence is determined by using rules. The sentence detection gives the structure, form and type of sentence.

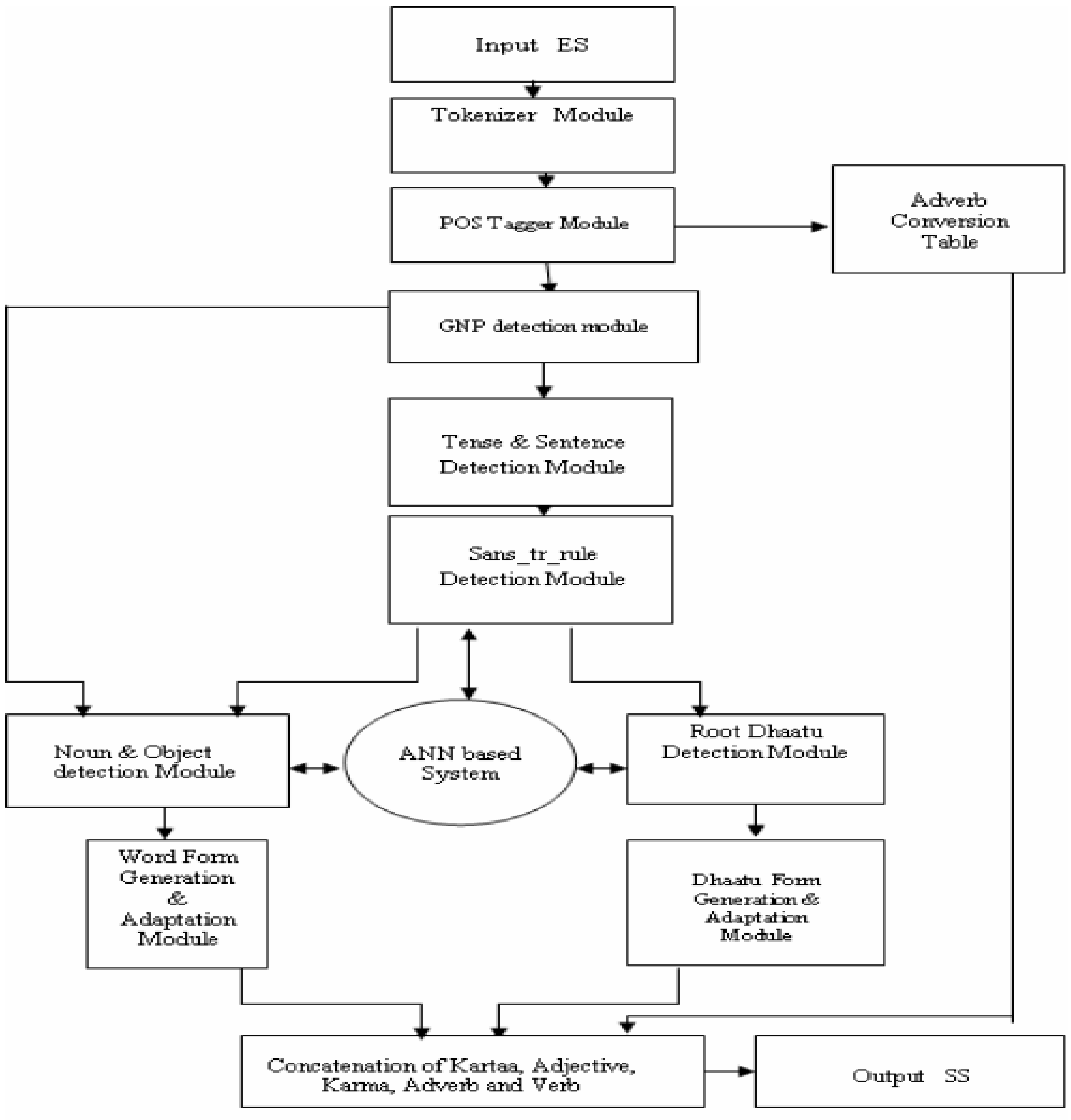

Input ES
Tokenizer Module
POS Tagger Module
Adverb Conversion Table
GNP detection module
Tense & Sentence Detection Module
Sans_tr_rule Detection Module
Noun & Object detection Module
ANN based System
Root Dhaatu Detection Module
Word Form Generation & Adaptation Module
Dhaatu Form Generation & Adaptation Module
Concatenation of Kartaa, Adjective, Karma, Adverb and Verb
Output SS

Figure1 . Information Flow in EST Model

3.5 Noun and Object Detection Module

This module gives noun for Sanskrit of the equivalent English noun. It uses ANN method for the selection of noun for Sanskrit. The adaptation rules are used to generate the word form.

3.6 Root Dhaatu Detection Module

This module gives verb for Sanskrit of the equivalent English verb. It uses ANN method for the selection of verb for Sanskrit. We apply adaptation rules to generate the required dhaatu form.

3.7 Sans_Tr_Rule Detection Module

This module gives the number of modules that is used in the Sanskrit translation. In this, we make input data and corresponding output data. The input data has structure, form and type of English sentence in the decimal coded form.

3.8 Adverb Conversion Table

We have stored fifty adverbs for Sanskrit of the equivalent English adverb in a database file and a fragment of this database file is shown in the table IV which has one to one correspondence.

Table IV: A Fragment of Adverb Conversion Table

S. No.	Adverb in English	Adverb in Sanskrit
1	Slowly	*Shanaih*
2	Suddenly	*Akasmaat*
3	Everywhere	*Sarvatra*
4	Continuous	*Anisham*
5	Fast	*Drutah*
6	Always	*Sadaa*
7	Today	*Adah*
8	Daily	*Pratidinam*

3.9 ANN Based Model

In the ANN based model, we use feed forward ANN for the selection of equivalent Sanskrit word such as noun (subject or object) and verb of English sentence. In feed forward ANN, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. The use of feed forward ANN overcomes the output vector limitation (Thrun, S.B., 1991). Our

motivation behind use of feed forward ANN in language processing tasks are work of Nemec, Peter (2003) and Khalilov, Maxim et al. (2008).
We basically perform three steps in ANN based system such as encoding of User Data Vector (UDV), input-output generation of UDV and decoding of UDV. The name of our data sets have called UDV here, which is used in feed forward ANN for the selection of equivalent Sanskrit word such as noun (subject or object) and verb of English sentence (Mishra, Vimal and Mishra, R. B., 2010b).

3.10 Encoding Of UDV

English alphabet consists of twenty-six characters which can be represented by five bit binary (2^5 =32, it ranges from 00000 to 11111). First, we write alphabet (a-z) into five bit binary in which alphabet "a" as 00001, to avoid the problem of divide by zero and alphabet "z" as 11010. For the training into ANN system, we make the alphabet to decimal coded form which is obtained by dividing each to thirty-two. This gives us input word in decimal coded form and output in corresponding Sanskrit word in roman script as encoded on the basis of table V.

Table V: Encoding of English Alphabet

S. No	**Character of alphabet**	**5-bit binary**	**Decimal of binary form divide by 32**
1	A	00001	0.031
2	B	00010	0.063
3	C	00011	0.094
.	.	.	.
.	.	.	.
25	Y	11001	0.781
26	Z	11010	0.813

The encoding of UDV rules are given below in algorithm I.

Algorithm I

Step1 Represent the English alphabet by 5 bit binary, such as assigning "a" as 00001 and "z" as 11010.
Step2 Divide each values of alphabet by 2^5 (32) and convert them into decimal coded form.
Step3 Make separate UDV for noun (subject or object) and verb that have one-to-one correspondence between noun (subject or object) and verb of English as input and Sanskrit as output.
Step4 Each input of UDV has five data values.

In the Encoding of UDV, we provide the data values to the noun (subject or object) and verb. In case of UDV for verb, UDV have five data values, say x1, x2, x3, x4 and x5 for input in English and six data values, say y1, y2, y3, y4 ,y5 and y6 for output in Sanskrit. In case of UDV for noun, UDV have five data values, say x1, x2, x3, x4 and x5 for input in English and seven data values, say y1, y2, y3, y4 ,y5, y6 and y7 for output in Sanskrit. For example, Man ([0.406 0.031 0.438 0.008 0.009]) as input to English noun and Nara([0.438 0.031 0.563 0.031 0.009 0.008 0.009]) as output to Sanskrit noun is encoded as shown in table VII. Similarly, Play ([0.500 0.375 0.031 0.781 0.009]) as input to English verb and Krid ([0.344 0.563 0.281 0.125 0.009 0.008]) as output to Sanskrit dhaatu is encoded as shown in table VI. The UDV for verb and noun are shown in table VI and table VII, respectively.

3.11 Input-Output Generation of UDV

The table VI and table VII shows the input-output pair of data for the two to five characters verb and noun in English as input and corresponding verb and noun in Sanskrit as output. We prepare UDV of noun (subject or object) and verb which is of maximum of five characters. In case of two characters noun or verb, we add three dummy values which range between 0.007 and 0.009, as suffix to UDV to make them five characters noun or verb. Similarly, we add two and one dummy values for three and four characters noun or verb etc respectively to make them five characters noun or verb.

After preparing the UDV, we train the UDV through feed forward ANN and then test the UDV. We get the output of Sanskrit word in the UDV form.

Table VI: UDV for Verb

S No	**Input English verb [x1 x2 x3 x4 x5]**	**Output Sanskrit dhaatu [y1 y2 y3 y4 y5 y6]**
1	Do [0.125 0.469 0.009 0.008 .007]	Kri [0.344 0.563 0.281 0.008 0.007 0.008]
2	Go [0.219 0.469 0.009 0.008 0.007]	Gam [0.219 0.031 0.406 0.008 0.007 0.009]
3	Ask [0.031 0.594 0.344 0.008 0.009]	Prach [0.500 0.563 0.031 0.094 0.250 0.008]
4	Eat [0.156 0.031 0.625 0.008 0.009]	Khaad [0.344 0.250 0.031 0.031 0.125 0.008]
5	Play [0.500 0.375 0.031 0.781 0.009]	Krid [0.344 0.563 0.281 0.125 0.009 0.008]

Table VII: UDV for Noun

S No	Input English noun [x1 x2 x3 x4 x5]	Output Sanskrit kartaa [y1 y2 y3 y4 y5 y6 y7]
1	Man [0.406 0.031 0.438 0.008 0.009]	Nara [0.438 0.031 0.563 0.031 0.009 0.008 0.009]
2	Ant [0.031 0.438 0.625 0.008 0.009]	Vamra [0.688 0.031 0.406 0.563 0.031 0.009 0.008]
3	City [0.094 0.281 0.625 0.781 0.009]	Nagar [0.438 0.031 0.219 0.031 0.563 0.009 0.008]
4	Hate [0.250 0.031 0.625 0.156 0.009]	Dvesa [0.125 0.688 0.156 0.594 0.031 0.009 0.008]
5	Pride [0.500 0.563 0.281 0.125 0.156]	Garva [0.219 0.031 0.563 0.688 0.031 0.009 0.008]

3.12 Decoding of UDV

The output given by ANN model is in decimal coded form. From the table VI, each values of a data set is compared with the values of column 4 in table V, one by one and the values with minimum difference is taken with its corresponding alphabet from the column 2 in table V. The rules are described for the decoding of UDV for verb that is given below in algorithm II.

Algorithm II

```
Step1   Input the decimal coded form of UDV in 5 double type variables, say x1, x2, x3, x4 and x5 and take corresponding output of the decimal coded form of UDV in 5 double type variables, say y1, y2, y3, y4, y5 and y6.
Step2   Compare y1 with each values of column 4 in table V.
Step3   If y1 match with value of column 4 in table V then select corresponding alphabet in table V.
Step4   If y1 does not match with value of column 4 in table V then make difference of y1 with all values of column 4 in table V.
Step5   Take the corresponding alphabet of minimum difference.
Step6   Repeat step 2 to 5 for y2, y3, y4, y5 and y6.
Step7   Put the corresponding alphabet of y1, y2, y3, y4, y5 and y6 into a string which is the desired decoded form of dhaatu in Sanskrit.
```

Similarly, we have developed rules for the decoding of UDV for noun. We have generated verb form and word form using rules.

3.13 Rules for Adaptation Processes of Parsmaipada Dhaatu

In Sanskrit, the verb used for handling the active voice of English sentence is Parsmipada dhaatu while Aatmaneypada dhaatu is used for handling the passive voice of English sentence. The adaptation processes of Parsmaipada Dhaatu are shown below in table VIII.

Table VIII: The Rule Set for Verb Form Generation of Parsmaipada Dhaatu: Present tense and Imperative Mood

Person	Present tense			Imperative		
	Singular	Dual	Plural	Singular	Dual	Plural
1st	Ati	atah	anti	atu	ataam	antu
2nd	Asi	athah	atha	a	atam	ata
3rd	Aami	aavah	aamah	aani	aava	aama

3.14 Rules for the Adaptation Processes of akAranta Word

A word that ends with characters "a" is called akAranta word such as the word "raama". The adaptation processes of akAranta word are shown below in table IX.

Table IX: The Rule Set for Word Form Generation of akAranta word: Masculine and Neuter Gender

Vibhakti	Masculine word			Neuter word		
	Singular	Dual	Plural	Singular	Dual	Plural
Nominative	h	au	ah	m	e	aani
Vocative	-	au	ah	-	e	aani
Accusative	am	au	an	m	e	aani
Instrumental	en	abhyaam	aih	en	abhyaam	aih
Dative	aay	abhyaam	ebhyah	aay	abhyaam	ebhyah
Ablative	aat	abhyaam	ebhyah	aat	abhyaam	ebhyah
Genitive	asya	ayoh	anaam	asya	ayoh	anaam

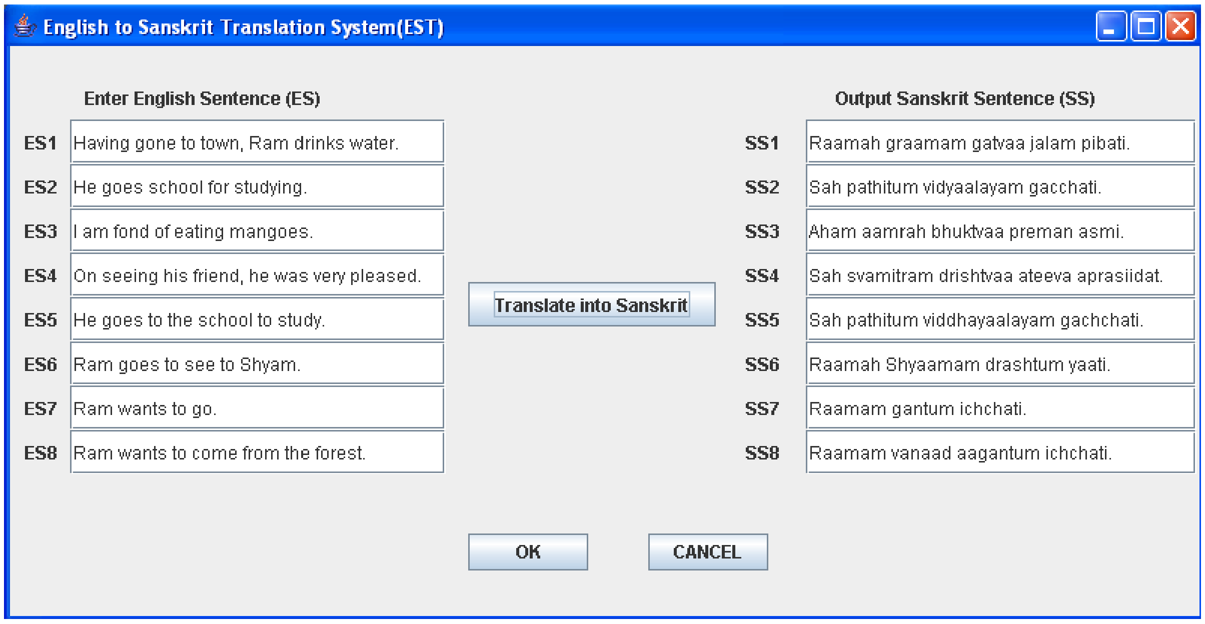
Locative	e	ayoh	esu	e	ayoh	esu

Similarly, we have adapted all types of word with all gender.

4. Implementation and Results

Our EST system has been implemented on windows platform using Java. The ANN model is implemented using MATLAB 7.1 neural Networks tool. We use feed forward ANN that gives matching of equivalent Sanskrit word of English word which handles noun and verb. We have a data set of 250 input-output pair for verb. The input, hidden and output values for verb is taken 5, 38 and 6. The training is terminated at a training error of 10^{-3} after 300 epochs. For the noun, we have 250 input-output pair in which the input, hidden and output values are taken 5, 15 and 7. This training is terminated at a training error of 10^{-2} after 300 epochs.

The result from our EST system for non-finite verbs type of English sentence is shown in figure 2.



	Enter English Sentence (ES)		Output Sanskrit Sentence (SS)
ES1	Having gone to town, Ram drinks water.	SS1	Raamah graamam gatvaa jalam pibati.
ES2	He goes school for studying.	SS2	Sah pathitum vidyaalayam gacchati.
ES3	I am fond of eating mangoes.	SS3	Aham aamrah bhuktvaa preman asmi.
ES4	On seeing his friend, he was very pleased.	SS4	Sah svamitram drishtvaa ateeva aprasiidat.
ES5	He goes to the school to study.	SS5	Sah pathitum viddhayaalayam gachchati.
ES6	Ram goes to see to Shyam.	SS6	Raamah Shyaamam drashtum yaati.
ES7	Ram wants to go.	SS7	Raamam gantum ichchati.
ES8	Ram wants to come from the forest.	SS8	Raamam vanaad aagantum ichchati.

Figure 2: Non-Finite Verbs Type of English Sentence

5. Evaluations

We evaluate the performance of our EST system that handle non-finite verbs of English sentences (ES) using different MT evaluation methods such as BLEU (BiLingual Evaluation Understudy) (Papineni et al., 2002), unigram Precision (P), unigram Recall (R), F-measure (F) (Melamed, R. et al., 2003) and METEOR (M) (Banerjee, S. et al., 2005) score. The evaluation scores of our EST system are encouraging which are calculated among 15 randomly selected sentences with our EST system (C) including reference translations (R) that are given below (Mishra & Mishra., 2009; 2010a).

1 ES: He goes to the school to study.
C: Sah pathitum viddhayaalayam gachchati.
R: Sah pathitum viddhayaalayam gataah.
2 ES: Ram goes to see to Shyam.
C: Raamah Shyaamam drashtum yaati.
R: Raamah Shyaamam ichchatum yaati.
3 ES: Ram wants to go.
C: Raamam gantum ichchati.
R: Raamam gantum ichchati.
4 ES: Ram wants to come from the forest.
C: Raamam vanaad aagantum ichchati.
R: Raamam vanaam aagantum ichchati.
5 ES: The book can be read by the boy.
C: Baalen pustakam pathitum shakyate.
R: Baalaken pustakam pathitum shakyate.
6 ES: I am fond of eating mango.
C: Aham aamram bhuktvaa preman asmi.
R: Aham aamram gajadhvaa preman asmi.
7 ES: Having gone to town, Ram drinks water.
C: Ramah gramam gatvaa jalam pibati.
R: Ramah gramam itvaa jalam pibati.
8 ES: I like reading book.
C: Aham pathitum pustakam vanaami.
R: Aham pathitum pustakam ichchati.
9 ES: After finishing his work, the boy went to school.
C: Baalakah svakaasyam kritvaa vidyaalayam agacchat.
R: Baalah svakaasyam kritvaa vidyaalayam agacchat.
10 ES: On seeing his friend, he was very pleased.
C: Sah svamitram drishtvaa ateeva aprasiidat.
R: Sah svamitram niriisya ateeva aprasiidat.
11 ES: Drinking of milk is always useful.
C: Duddh paanam sarvada hitkaram bhavati.
R: Duddh paanam sarvada labhapradam bhavati.
12 ES: Sleeping is necessary for life.
C: Svapnam jiivansya nitaantam asti.
R: Svapnam jiivansya aavasyakam asti.
13 S: He goes school for studying.
C: Sah pathitum vidyaalayam gacchati.
R: Sah pathitum paathashaalam gacchati.
14 ES: Playing cricket makes us active.
C: Cricket kridanam usman kriyasheelam nirmati.
R: Cricket kridanam usman gatisheelam nirmati.

15 ES: Reading of book is useful.
C: Pustak pathnam hitkaram asti.
R: Pustak pathnam labhapradam asti.

The evaluation scores for fifteen randomly selected non-finite verbs type of English sentences are shown in table X.

Table X: Performance Evaluation Scores for Non-Finite Verbs Type of English Sentences in our EST System

S. No.	BLEU	P	R	F	M
1	0.48	0.75	0.75	0.75	0.7516
2	0.2708	0.75	0.75	0.75	0.7636
3	1.0	1.0	0.75	0.75	0.7516
4	0.2708	0.75	0.75	0.75	0.7827
5	0.4792	0.750	0.750	0.750	0.7540
6	0.325	0.8	0.8	0.8	0.811
7	0.325	0.8	0.8	0.8	0.811
8	0.48	0.75	0.75	0.75	0.7516
9	0.6717	0.8	0.8	0.8	0.811
10	0.325	0.8	0.8	0.8	0.811
11	0.325	0.8	0.8	0.8	0.8057
12	0.2708	0.750	0.750	0.750	0.7827
13	0.2708	0.750	0.750	0.750	0.7827
14	0.325	0.8	0.8	0.8	0.8057
15	0.2708	0.750	0.750	0.750	0.7827

6. Conclusions

Our paper describes the handling of non-finite verbs in English to Sanskrit machine translation. We have proposed a novel method that is basically comprised the combination of two approaches: rule based model and the dictionary matching by ANN model for the handling of non-finite verbs in English to Sanskrit machine translation. The rule based model enhances the adaptation process. The results from our system are encouraging.

References

1. Banerjee, S. and Alon, Lavie. 2005. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. In Proceedings of Workshop on "Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization", ACL 2005. pp 65-73. Ann Arbor, Michigan.

2. Briggs, Rick. 1985. Knowledge Representation in Sanskrit and Artificial Intelligence. The AI Magazine, pp 33–39.
3. Egenes, Thomas. 2000. *Introduction to Sanskrit, Part one & two.* Motilal Banarasidas Publishers Pvt. Ltd., First edn.
4. Goyal, Pawan and Sinha, R.M.K. 2009. A Study towards Design of English to Sanskrit Machine Translation System. Lecture notes in computer science sub series: Lecture notes in artificial intelligence, Springer- Verlag. Vol. 5402, pp 287-305.
5. Hossain, G. M. 2008. A brief introduction to Anubadok the Bengali machine translator. pp 1-8.
6. Huet, G'erard. 2003. Towards Computational Processing of Sanskrit, Recent Advances in Natural Language Processing, Proceedings of the International Conference ICON, Mysore, India.
7. Jha, Girish N et al. 2006. Towards a Computational analysis system for Sanskrit, In Proceedings of first National symposium on Modeling and Shallow parsing of Indian Languages at Indian Institute of Technology Bombay, pp 25-34.
8. Kadambini K, Rama Sree R.J., Prof.Rama Krishnamacharyulu K.V. 2009. An English-Sanskrit Machine Translation using Link Parser. pp 1-7.
9. Khalilov, Maxim et al. 2008. Neural Network Language Models for Translation with Limited Data. In Proceedings of 20th IEEE International Conference on Tools with Artificial. pp 445-451.
10. Melamed, R. Green, and J. Turian. 2003. Precision and Recall of Machine Translation. In Proceedings of the HLTNAACL 2003. Short Papers: 61–63, Edmonton, Canada.
11. Mishra, Vimal and Mishra, R. B. 2008. Study of Example based English to Sanskrit Machine Translation. Journal of Research and Development in Computer Science and Engineering, "Polibits". Vol.37, pp 43-54.
12. Mishra, Vimal and Mishra, R. B., 2009. Evaluation of English to Sanskrit machine translation, Journal "ATTI Della Fondazione GIORGIO RONCHI", Vol. LXIV, No. 3, pp 345-353.
13. Mishra, Vimal and Mishra, R. B., 2010. ANN and Rule based model for English to Sanskrit Machine Translation. INFOCOMP Journal of Computer Science, Vol. 9(1), pp 80-89.
14. Mishra, Vimal and Mishra, R. B. 2010a. Performance Evaluation of English to Sanskrit Machine Translation System: A Novel Approach, In Proceedings of International Joint Conference on Information and Communication Technology, 9th-10th January, 2010, pp 181-186, IIMT, Bhubaneswar, India.
15. Nautiyal, Chakradhar. 1997. Vrihad Anuvaad Chandrika. 4th Ed., Motilal Banarasidas Publishers Pvt. Ltd.
16. Nemec, Peter, 2003. Application of Artificial Neural Networks in Morphological Tagging of Czech, pp 1-8.

17. Papineni, Kishore & Roukos, Salim et al. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics, pp 1– 9.
18. Ramanujan, P. 2000. Computer Processing of Sanskrit, In Proceedings of CALP-2, I.I.T. Kanpur, India. pp 1- 10.
19. Thrun, S.B. 1991. The MONK's Problems: a performance comparison of different learning algorithms, Technical Report CMU-CS-91-197, Carnegie Mellon University.
20. Wren, P.C. and Martin, H. 1994. High School English Grammar and Composition, S. Chand and Company Publishers Ltd., New Delhi, India.