IL-ILMT SAMPARK: A Hybrid Machine Translation system

Christopher[1] M, Uma Maheshwar Rao[2] G
[1]LTRC, [2]Center for Applied Linguistic and Translation Studies
[1]IIIT-Hyderabad, [2]University of Hyderabad
{[1]efthachris, [2]gurao}@gmail.com

1 Introduction

The development of Machine Translation (MT) is one of the most challenging tasks of Natural Language Processing Applications. In MT there are a number of methods that are being practiced all over the world, chiefly, they are Direct Method, Interlingual Methods, Transfer Based Methods and a combination of these beside the statistical and corpus based methods. It is known fact that Indian languages exhibit a considerable amount of diversity between them at every level viz. morphological, syntactic, semantic and lexical levels. In the Transfer Based Method a representation of source language (SL) at certain level is transfered to the corresponding target language (TL) representation. Keeping these in mind, building a Machine Translation System for these languages using Transfer based Method can be non-trivial and challenging. The present paper discusses the successful implementation of the Transfer Based Approach to the Machine Translation (MT) System for some of the Indian languages like Hindi<->{Telugu, Tamil, Punjabi, Marathi, Bengali, Urdu, Kannada} and Tamil<->Telugu, Malayalam<->Tamil. These systems were built at 11 different institution across India.

This system (IL-ILMT[1]) is an assembly of various linguistic modules run on specific engines whose output is sequentially maneuvered and modified by a series of operations till the output is generated. The most crucial linguistic modules include, Morphological Analyzer (MA), Parts of Speech Tagger (POS-T), Simple Parser (SP), Transfer Grammar Component (TG), Lexical Transfer module consisting of a Bilingual Dictionary and a Concept Dictionary, Agreement module (AGR) and a Morphological Generator (Wordgen). The system is built and is now being tested and evaluated at *www.ilmt.iiit.ac.in/sampark*.

In this paper, we report the ILMT-SAMPARK MT system. In Sect. 2, we describe its architecture and certain main concepts. Sect. 3 concerns with the implementation and description of various modules. Finally, we conclude our paper in Sect. 4 .

2 ILMT- SAMPARK Architecture :

The IL-ILMT *Sampark* MT system is based on analyze-transfer-generate paradigm. First, analysis of the source language would be done, then a transfer of analyzed structures and vocabulary to the target language would be carried out, and finally the target language would be generated. For this we have adopted the Black Board Architecture as it enables the re-use the output of the previous modules. In this architecture, heterogeneous nature of modules do not effect their operations as all of them operate on common in-memory data structure. In the Black Board Architecture there is no fixed order of module execution.

The black board architecture gives a flexible environment for module integration where each module is pluggable component, and one module can be easily replaced by another version and continues to function even after a module in between has failed. The following modules in the pipe may still work on the existing in-memory data structure, and operations can proceed further providing graceful degradation of the system.

3. Module Level details of the System

3.1. The Format:

The entire system works on a unique standard format called the *Shakti Standard Format (SSF)* (Akshra Bharati et.al, 2009). The *multi-column* format vividly represents input and output of each module throughout the system. This is especially designed to represent different levels of analysis. The analyses are : 1. Constituent level analysis and 2. Relational-Structure level analysis. The former is used to store simple phrase level analysis and the latter for storing relations between simple parses. Feature structures are used to store attribute-value pairs for a phrasal node as well as for a word or a token. The attribute value pairs store relations in different columns. The following is a description of the column format in SSF:

Column 1 stores the node address, mainly for human readability. Column 2 stores the word or word-group input. The symbol "((" represents the start of the word or word-group and the symbol "))" to represent the end of the word or word-group. Column 3 stores the chunk name or the POS tag of the words occurring in the sentence. Column 4 stores the Morphological information (feature structures) of the words. Column 4 contains attribute features of word form in 8 fields, which are mandatory for all languages. These fields are as follows: 1st field contains root/base of the word form, 2nd field contains lexical category of word form, 3rd , 4th and 5th contains values for gender, number and person respectively, 6th field stores information with regard to the oblique or direct form of the stem in case of nouns, 7th field contains case marker in case of nouns and tense in case of verbs and the 8th field stores the exact suffix representing the features presented in 3-7. The other feature about the word is stored in the Column 4 as key value pair.

***3.2. Source Analysis**:*

Tokenizer: The tokenizer converts a text into a sequence of tokens (words, punctuation marks, etc.) within the Shakti Standard Format.

Input: *prAkqwika suMxarawA se owaprowa OlI BArawa kA prasixXa paryataka sWala hE.*

Output:

1 prAkqwika
2 suMxarawA
3 se
4 owaprowa
5 OlI
6 BArawa
7 kA
8 prasixXa
9 paryataka
10 sWala
11 hE
12 .

a. Morphological analyzer (MA): A Morphological Analyzer analyzes and identifies the root and the grammatical features of the word. Word and Paradigm based approaches have given good success rates for Indian languages and the current Morphological Analyzer adopted this model. (Uma Maheshwar Rao et. al. 2007). The computational module used in this MT system currently provides slots for 8 feature values for each word, viz. root, lexical category, gender, number, person, case, case marker or TAM and suffix. The lexical categories are nine:- they are noun (n) , verb (v), adjective (adj), pronoun (pn), adverb (adv), postposition (psp), number (num), nouns of space and time (NST) and indeclinable (avy) (Uma Maheshwar Rao et.al. 2007).

1 prAkqwika unk <fs af='prAkqwika,adj,any,any,,any,,'>
2 suMxarawA unk <fs af='suMxarawA,n,f,sg,3,d,0,0'>|<fs af='suMxarawA,n,f,pl,3,d,0,0'>|<fs af='suMxarawA,n,f,sg,3,o,0,0'>|<fs af='suMxarawA,n,f,pl,3,o,0,0'>
3 se unk <fs af='se,v,any,any,any,,0,0'>|<fs af='se,psp,,,,,,'>|<fs af='sA,psp,m,sg,,o,kA,kA'>|<fs af='sA,psp,m,pl,,d,kA,kA'>|<fs af='sA,psp,m,pl,,o,kA,kA'>
4 owaprowa unk <fs af='owaprowa,adj,any,any,,any,,'>
5 OlI unk <fs af='OlI,unk,,,,,,'>
6 BArawa unk <fs af='BArawa,n,m,sg,3,d,0,0'>|<fs af='BArawa,n,m,pl,3,d,0,0'>|<fs af='BArawa,n,m,sg,3,o,0,0'>|<fs af='BArawa,n,m,pl,3,o,0,0'>
7 kA unk <fs af='kA,psp,m,sg,,d,kA,kA'>
8 prasixXa unk <fs af='prasixXa,adj,any,any,,any,,'>|<fs af='prasixXa,n,m,sg,3,d,0,0'>|<fs af='prasixXa,n,m,pl,3,d,0,0'>|<fs af='prasixXa,n,m,sg,3,o,0,0'>|<fs af='prasixXa,n,m,pl,3,o,0,0'>
9 paryataka unk <fs af='paryataka,n,m,sg,3,d,0,0'>|<fs af='paryataka,n,m,pl,3,d,0,0'>|<fs af='paryataka,n,m,sg,3,o,0,0'>|<fs af='paryataka,adj,any,any,,any,,'>
10 sWala unk <fs af='sWala,n,m,sg,3,d,0,0'>|<fs af='sWala,n,m,pl,3,d,0,0'>|<fs af='sWala,n,m,sg,3,o,0,0'>
11 hE unk <fs af='hE,v,any,sg,2,,hE,hE'>|<fs af='hE,v,any,sg,3,,hE,hE'>
12 . unk <fs af='.,punc,,,,,,'>

b. Parts of speech tagger (POS-T): Part of speech tagging is the process of assigning a

unique part of speech to each word (token) in the sentence. This process helps in identifying the role of each word (token) in a sentence. There are a number of approaches, such as rule-based, statistics based, transformation-based etc. which are used for POS tagging. Here we have implemented a statistics based tagger which uses the *GOLD* standard (General Ontological Linguistic Data) tagged text data for the extractor of rules for future use on the input data text (following ILMT Tagset, ILMT 2007).

```
1	prAkqwika	JJ	<fs af='prAkqwika,adj,any,any,,any,,'>
2	suMxarawA	NN	<fs af='suMxarawA,n,f,sg,3,d,0,0'>|<fs af='suMxarawA,n,f,pl,3,d,0,0'>|<fs af='suMxarawA,n,f,sg,3,o,0,0'>|<fs af='suMxarawA,n,f,pl,3,o,0,0'>
3	se	PSP	<fs af='se,v,any,any,any,,0,0'>|<fs af='se,psp,,,,,,'>|<fs af='sA,psp,m,sg,,o,kA,kA'>|<fs af='sA,psp,m,pl,,d,kA,kA'>|<fs af='sA,psp,m,pl,,o,kA,kA'>
4	owaprowa	JJ	<fs af='owaprowa,adj,any,any,,any,,'>
5	OIl	NNP	<fs af='OIl,unk,,,,,,'>
6	BArawa	NNP	<fs af='BArawa,n,m,sg,3,d,0,0'>|<fs af='BArawa,n,m,pl,3,d,0,0'>|<fs af='BArawa,n,m,sg,3,o,0,0'>|<fs af='BArawa,n,m,pl,3,o,0,0'>
7	kA	PSP	<fs af='kA,psp,m,sg,,d,kA,kA'>
8	prasixXaJJ	<fs af='prasixXa,adj,any,any,,any,,'>|<fs af='prasixXa,n,m,sg,3,d,0,0'>|<fs af='prasixXa,n,m,pl,3,d,0,0'>|<fs af='prasixXa,n,m,sg,3,o,0,0'>|<fs af='prasixXa,n,m,pl,3,o,0,0'>
9	paryataka	NN	<fs af='paryataka,n,m,sg,3,d,0,0'>|<fs af='paryataka,n,m,pl,3,d,0,0'>|<fs af='paryataka,n,m,sg,3,o,0,0'>|<fs af='paryataka,adj,any,any,,any,,'>
10	sWala	NN	<fs af='sWala,n,m,sg,3,d,0,0'>|<fs af='sWala,n,m,pl,3,d,0,0'>|<fs af='sWala,n,m,sg,3,o,0,0'>
11	hE	VM	<fs af='hE,v,any,sg,2,,hE,hE'>|<fs af='hE,v,any,sg,3,,hE,hE'>
12	.	SYM	<fs af='.,punc,,,,,,'>
```

c. Chunker: Chunking involves identifying non-recursive combinations of word groups involving nouns (NP), verbs (VGF/VGNF), adjectives (JJP) and adverbs (RBP) etc. in a given sentence. Here we use statistical methods to identify and tag chunks in a sentence (following ILMT Tagset, ILMT 2007).

```
1	((	NP
1.1	prAkqwika	JJ	<fs af='prAkqwika,adj,any,any,,any,,'>
1.2	suMxarawA	NN	<fs af='suMxarawA,n,f,sg,3,d,0,0'>
1.3	se	PSP	<fs af='se,psp,,,,,,'>
	))
2	((	NP
2.1	owaprowa	JJ	<fs af='owaprowa,adj,any,any,,any,,'>
2.2	OIl	NNP	<fs af='OIl,unk,,,,,,' poslcat="NM">
	))
3	((	NP
3.1	BArawa	NNP	<fs af='BArawa,n,m,sg,3,d,0,0'>
3.2	kA	PSP	<fs af='kA,psp,m,sg,,d,kA,kA'>
	))
4	((	NP
4.1	prasixXaJJ	<fs af='prasixXa,adj,any,any,,any,,'>
4.2	paryataka	NN	<fs af='paryataka,n,m,sg,3,d,0,0'>
	))
5	((	NP
5.1	sWala	NN	<fs af='sWala,n,m,sg,3,d,0,0'>
	))
6	((	VGF
6.1	hE	VM	<fs af='hE,v,any,sg,3,,hE,hE'>
6.2	.	SYM	<fs af='.,punc,,,,,,' poslcat="NM">
	))
```

d. Named Entity Recognizer (NER): The identification, recognition and tagging of proper nouns such as names of persons and organizations (ILMT 2007) is achieved by this module.

```
1	((	NP
1.1	prAkqwika	JJ	<fs af='prAkqwika,adj,any,any,,any,,'>
1.2	suMxarawA	NN	<fs af='suMxarawA,n,f,sg,3,d,0,0'>
1.3	se	PSP	<fs af='se,psp,,,,,,'>
	))
2	((	NP
2.1	owaprowa	JJ	<fs af='owaprowa,adj,any,any,,any,,'>
```

```
2.2     OlI     NNP     <fs af='OlI,unk,,,,,,' poslcat="NM">
        ))
3       ((      NP
3.1     BArawa  NNP     <fs af='BArawa,n,m,sg,3,d,0,0'>
3.2     kA      PSP     <fs af='kA,psp,m,sg,,d,kA,kA'>
        ))
4       ((      NP
4.1     prasixXa        JJ      <fs af='prasixXa,adj,any,any,,any,,'>
4.2     paryataka       NN      <fs af='paryataka,n,m,sg,3,d,0,0'>
        ))
5       ((      NP
5.1     sWala   NN      <fs af='sWala,n,m,sg,3,d,0,0'>
        ))
6       ((      VGF
6.1     hE      VM      <fs af='hE,v,any,sg,3,,hE,hE'>
6.2     .       SYM     <fs af='.,punc,,,,,,' poslcat="NM">
        ))
```

e. Simple parser (SP): Identifies and names Thematic relations between a verb and its participant noun in the sentence, based on the Computational Paninian Grammar framework (Akshar Bharati et.al 2009).

```
1       ((      NP      <fs af='suMxarawA,n,f,sg,3,d,0_se,0' head="suMxarawA" drel=k3:6 name=1>
1.1     prAkqwika       JJ      <fs af='prAkqwika,adj,any,any,any,any,,'>
1.2     suMxarawA       NN      <fs af='suMxarawA,n,f,sg,3,d,0,0' name="suMxarawA">
        ))
2       ((      NP      <fs af='OlI,n,n,sg,3,,0,0' head="OlI" drel=k1:6 name=2 poslcat="NM">
2.1     owaprowa        JJ      <fs af='owaprowa,adj,any,any,,any,,'>
2.2     OlI     NNP     <fs af='OlI,n,n,sg,3,,0,0' poslcat="NM" name="OlI">
        ))
3       ((      NP      <fs af='BArawa,n,m,sg,3,d,0_kA,0' head="BArawa" drel=r6:4 name=3>
3.1     BArawa  NNP     <fs af='BArawa,n,m,sg,3,d,0,0' name="BArawa">
        ))
4       ((      NP      <fs af='paryataka,n,m,sg,3,d,0,0' head="paryataka" name=4>
4.1     prasixXa        JJ      <fs af='prasixXa,adj,any,any,any,any,,'>
4.2     paryataka       NN      <fs af='paryataka,n,m,sg,3,d,0,0' name="paryataka">
        ))
5       ((      NP      <fs af='sWala,n,m,sg,3,d,0,0' head="sWala" name=5>
5.1     sWala   NN      <fs af='sWala,n,m,sg,3,d,0,0' name="sWala">
        ))
6       ((      VGF     <fs af='hE,v,any,sg,3,,hE,hE' head="hE" name=6>
6.1     hE      VM      <fs af='hE,v,any,sg,3,,hE,hE' name="hE">
6.2     .       SYM     <fs af='.,punc,,,,,,' poslcat="NM">
        ))
```

f. Transfer Grammar (TG): Wherever, the source language does not have an equivalent structure in the target language, a structural transformation is required to convert the source language structure into an acceptable target language structure. Such cases can be found at every level wherever divergence occurs between the source and the target language. TG Module contains rules which convert the parsed structure of the source language into the desired structure in the target language giving the acceptable target structures.

```
1       ((      NP      <fs af='suMxarawA,n,f,sg,3,d,0_se,0' head="suMxarawA" drel=k3:6 name=1>
1.1     prAkqwika       JJ      <fs af='prAkqwika,adj,any,any,any,any,,'>
1.2     suMxarawA       NN      <fs af='suMxarawA,n,f,sg,3,d,0,0' name="suMxarawA">
        ))
2       ((      NP      <fs af='OlI,n,n,sg,3,,0,0' head="OlI" drel=k1:6 name=2 poslcat="NM">
2.1     owaprowa        JJ      <fs af='owaprowa,adj,any,any,,any,,'>
2.2     OlI     NNP     <fs af='OlI,n,n,sg,3,,0,0' poslcat="NM" name="OlI">
        ))
3       ((      NP      <fs af='BArawa,n,m,sg,3,d,0_kA,0' head="BArawa" drel=r6:4 name=3>
3.1     BArawa  NNP     <fs af='BArawa,n,m,sg,3,d,0,0' name="BArawa">
        ))
4       ((      NP      <fs af='paryataka,n,m,sg,3,d,0,0' head="paryataka" name=4>
4.1     prasixXa        JJ      <fs af='prasixXa,adj,any,any,any,any,,'>
4.2     paryataka       NN      <fs af='paryataka,n,m,sg,3,d,0,0' name="paryataka">
```

```
        ))
5       ((      NP      <fs af='sWala,n,m,sg,3,d,0,0' head="sWala" name=5>
5.1     sWala   NN      <fs af='sWala,n,m,sg,3,d,0,0' name="sWala">
        ))
6       ((      VGF     <fs af='hE,v,any,sg,3,,yA,hE' head="hE" name=6>
6.1     hE      VM      <fs af='hE,v,any,sg,3,,hE,hE' name="hE">
6.2     .       SYM     <fs af='.,punc,,,,,,' poslcat="NM">
        ))
```

g. Multi-Word Expression Transfer (MWE): Multi-Word Expression module involves identification and transfer of frequently used non-compositional phrases, compounds, reduplicatives, etc. from the source language into the target language.

```
1       ((      NP      <fs af='suMxarawA,n,f,sg,3,d,0_se,0' head="suMxarawA" drel=k3:6 name=1>
1.1     prAkqwika       JJ      <fs af='prAkqwika,adj,any,any,any,any,,'>
1.2     suMxarawA       NN      <fs af='suMxarawA,n,f,sg,3,d,0,0' name="suMxarawA">
        ))
2       ((      NP      <fs af='Oll,n,n,sg,3,,0,0' head="Oll" drel=k1:6 name=2 poslcat="NM">
2.1     owaprowa        JJ      <fs af='owaprowa,adj,any,any,,any,,'>
2.2     Oll     NNP     <fs af='Oll,n,n,sg,3,,0,0' poslcat="NM" name="Oll">
        ))
3       ((      NP      <fs af='BArawa,n,m,sg,3,d,0_kA,0' head="BArawa" drel=r6:4 name=3>
3.1     BArawa  NNP     <fs af='BArawa,n,m,sg,3,d,0,0' name="BArawa">
        ))
4       ((      NP      <fs af='paryataka,n,m,sg,3,d,0,0' head="paryataka" name=4>
4.1     prasixXaJJ      <fs af='prasixXa,adj,any,any,any,any,,'>
4.2     -@paryAtaka     NN      <fs af='-@paryAtaka,n,m,sg,3,d,0,0' name="paryataka">
        ))
5       ((      NP      <fs af='sWala,n,m,sg,3,d,0,0' head="sWala" name=5>
5.1     ^@sWalaM        NN      <fs af='^@sWalaM,n,m,sg,3,d,0,0' name="sWala">
        ))
6       ((      VGF     <fs af='hE,v,any,sg,3,,yA,hE' head="hE" name=6>
6.1     hE      VM      <fs af='hE,v,any,sg,3,,hE,hE' name="hE">
6.2     .       SYM     <fs af='.,punc,,,,,,' poslcat="NM">
        ))
```

h. Lexical transfer (LT): Root words identified by the morphological analyzer are looked up in a bilingual dictionary for the target language equivalent using conceptuvally linked appropriate lexical substitution including function words.

```
1.1     ((      NP      <fs af='సౌందర్యం,n,f,sg,3,d,@తో,0' head='suMxarawA' drel='k3:6' name='1'>
1.1.1   ప్రాకృతిక  JJ      <fs af='ప్రాకృతిక,adj,any,any,any,any,,'>
1.1.2   సౌందర్యం  NN      <fs af='సౌందర్యం,n,f,sg,3,d,@0,0' name='suMxarawA'>
        ))
1.2     ((      NP      <fs af='@ఔలీ,n,n,sg,3,,@0,0' head='Oll' drel='k1:6' name='2' poslcat='NM'>
1.2.1   ప్రభావితమైన      JJ      <fs af='ప్రభావితమైన,adj,any,any,,any,,'>
1.2.2   @ఔలీ  NNP     <fs af='@ఔలీ,n,n,sg,3,,@0,0' name='Oll' poslcat='NM'>
        ))
1.3     ((      NP      <fs af='భారత్,n,m,sg,3,d,@యొక్క,0' head='BArawa' drel='r6:4' name='3'>
1.3.1   భారత్  NNP     <fs af='భారత్,n,m,sg,3,d,@0,0' name='BArawa'>
        ))
1.4     ((      NP      <fs af='పర్యాటకుడు,n,m,sg,3,d,@0,0' head='paryataka' name='4'>
1.4.1   ప్రసిద్ధ  JJ      <fs af='ప్రసిద్ధ,adj,any,any,any,any,,'>
1.4.2   @-paryAtaka     NN      <fs af='@-paryAtaka,n,m,sg,3,d,@0,0' name='paryataka'>
        ))
1.5     ((      NP      <fs af='స్థలం,n,m,sg,3,d,@0,0' head='sWala' name='5'>
1.5.1   sWalaM  NN      <fs af='sWalaM,n,m,sg,3,d,@0,0' name='sWala'>
        ))
```

1.6	((	VGF	<fs af='ఉండు,v,any,sg,3,,ఆ,hE' head='hE' name='6'>
1.6.1	ఉండు	VM	<fs af='ఉండు,v,any,sg,3,,ఉండు,hE' name='hE'>
1.6.2	.	SYM	<fs af='.,punc,,,,,,' poslcat='NM'>
	))		

i. Agreement (Agr): Performs checking and reconstructing gender-number-person agreement (wherever it is required) between the subject and the predicate in the target sentence, ensuring proper agreement. It is also called *Sentence Generator*.

1	((	NP	<fs af='సౌందర్యం,n,n,sg,3,d,తో,0' head= drel=k3:6 name=1>
1.1	ప్రాకృతిక	JJ	<fs af='ప్రాకృతిక,adj,,,,any,,'>
1.2	సౌందర్యం	NN	<fs af='సౌందర్యం,n,n,sg,3,d,0,0' name=suMxarawA>
	))		
2	((	NP	<fs af='ఔలీ,n,n,sg,3,,0,0' head=OlI drel=k1:6 name=2 poslcat=NM>
2.1	ప్రభావితమైన	JJ	<fs af='ప్రభావితమైన,adj,,,,any,,'>
2.2	ఔలీ	NNP	<fs af='ఔలీ,n,n,sg,3,,0,0' name=OlI poslcat=NM>
	))		
3	((	NP	<fs af='భారత్,n,n,sg,3,d,మొక్క,0' head=BArawa drel=r6:4 name=3>
3.1	భారత్	NNP	<fs af='భారత్,n,n,sg,3,d,0,0' name=BArawa>
	))		
4	((	NP	<fs af='పర్యాటకుడు,n,m,sg,3,d,0,0' head=paryataka name=4>
4.1	ప్రసిద్ధ	JJ	<fs af='ప్రసిద్ధ,adj,,,,any,,'>
4.2	@-పర్యాటక	NN	<fs af='@-పర్యాటక,n,n,sg,3,o,0,0' name=paryataka>
	))		
5	((	NP	<fs af='స్థలం,n,n,sg,3,d,0,0' head=sWala name=5>
5.1	స్థలం	NN	<fs af='స్థలం,n,n,sg,3,d,0,0' name=sWala>
	))		
6	((	VGF	<fs af='ఉండు,v,n,sg,3,,ఆ,hE' head= drel=head=hE name=6>
6.1	ఉండు	VM	<fs af='ఉండు,v,n,sg,3,,ఉండు,hE' head= drel=name=hE>
6.2	.	SYM	<fs af='.,punc,,,,,,' poslcat='NM'>
	))		

j. Vibhakti Splitter or Complex Inflection Splitter (VBS): Separates complex cases of inflections involving one or more postpositions in case of nouns or auxiliary verbs incase of verbs ensuring proper word generation.

1	((	NP	<fs af='సౌందర్యం,n,n,sg,3,d,తో,0' head= drel=k3:6 name=1>
1.1	ప్రాకృతిక	JJ	<fs af='ప్రాకృతిక,adj,,,,any,,'>
1.2	సౌందర్యం	NN	<fs af='సౌందర్యం,n,n,sg,3,d,తో,0' name=suMxarawA>
	))		
2	((	NP	<fs af='ఔలీ,n,n,sg,3,,0,0' head=OlI drel=k1:6 name=2 poslcat=NM>
2.1	ప్రభావితమైన	JJ	<fs af='ప్రభావితమైన,adj,,,,any,,'>
2.2	ఔలీ	NNP	<fs af='ఔలీ,n,n,sg,3,,0,0' name=OlI poslcat=NM>
	))		
3	((	NP	<fs af='భారత్,n,n,sg,3,d,మొక్క,0' head=BArawa drel=r6:4 name=3>
3.1	భారత్	NNP	<fs af='భారత్,n,n,sg,3,d,మొక్క,0' name=BArawa>
	))		
4	((	NP	<fs af='పర్యాటకుడు,n,m,sg,3,d,0,0' head=paryataka name=4>

```
4.1     ప్రసిద్ధ        JJ      <fs af='ప్రసిద్ధ,adj,,,,any,,'>
4.2     @-పర్యాటక              NN      <fs af='@-పర్యాటక,n,n,sg,3,o,0,0' name=paryataka>
        ))
5       ((              NP      <fs af='స్థలం,n,n,sg,3,d,0,0' head=sWala name=5>
5.1     స్థలం           NN      <fs af='స్థలం,n,n,sg,3,d,0,0' name=sWala>
        ))
6       ((              VGF     <fs af='ఉండు,v,n,sg,3,,ఆ,hE' head=drel=head=hE name=6>
6.1     ఉండు            VM      <fs af='ఉండు,v,n,sg,3,,ఆ,hE' head=drel=name=hE>
6.2     .               SYM     <fs af='.,punc,,,,,,' poslcat='NM'>
        ))
```

k. Word generator (WG): This module takes root words and their associated grammatical features, selects appropriate suffixes and concatenates them into well formed word forms.

```
1       ((              NP      <fs af='సౌందర్యం,n,n,sg,3,d,తో,0' head= drel=k3:6 name=1>
1.1     ప్రాకృతిక       JJ      <fs af='ప్రాకృతిక,adj,,,,any,,'>
1.2     సౌందర్యంతో             NN      <fs af='సౌందర్యం,n,n,sg,3,d,తో,0' name=suMxarawA>
        ))
2       ((              NP      <fs af='ఔలీ,n,n,sg,3,,0,0' head=Oli drel=k1:6 name=2 poslcat=NM>
2.1     ప్రభావితమైన            JJ      <fs af='ప్రభావితమైన,adj,,,,any,,'>
2.2     ఔలీ             NNP     <fs af='ఔలీ,n,n,sg,3,,0,0' name=Oli poslcat=NM>
        ))
3       ((              NP      <fs af='భారత్,n,n,sg,3,d,యొక్క,0' head=BArawa drel=r6:4 name=3>
3.1     భారత్యొక్క     NNP     <fs af='భారత్,n,n,sg,3,d,యొక్క,0' name=BArawa>
        ))
4       ((              NP      <fs af='పర్యాటకుడు,n,m,sg,3,d,0,0' head=paryataka name=4>
4.1     ప్రసిద్ధ        JJ      <fs af='ప్రసిద్ధ,adj,,,,any,,'>
4.2     పర్యాటక        NN      <fs af='@-పర్యాటక,n,n,sg,3,o,0,0' name=paryataka>
        ))
5       ((              NP      <fs af='స్థలం,n,n,sg,3,d,0,0' head=sWala name=5>
5.1     స్థలం           NN      <fs af='స్థలం,n,n,sg,3,d,0,0' name=sWala>
        ))
6       ((              VGF     <fs af='ఉండు,v,n,sg,3,,ఆ,hE' head=drel=head=hE name=6>
6.1     ఉంది            VM      <fs af='ఉండు,v,n,sg,3,,ఆ,hE' head=drel=name=hE>
6.2     .               SYM     <fs af='.,punc,,,,,,' poslcat='NM'>
        ))
```

***l. Post Processing*:** If necessary enables unacceptable sequences of words automatically modified by a set of post processing rules to generate more acceptable structures of the target language.

```
1       ((              NP      <fs af='సౌందర్యం,n,n,sg,3,d,తో,0' head= drel=k3:6 name=1>
1.1     ప్రాకృతిక       JJ      <fs af='ప్రాకృతిక,adj,,,,any,,'>
1.2     సౌందర్యంతో             NN      <fs af='సౌందర్యం,n,n,sg,3,d,తో,0' name=suMxarawA>
        ))
2       ((              NP      <fs af='ఔలీ,n,n,sg,3,,0,0' head=Oli drel=k1:6 name=2 poslcat=NM>
2.1     ప్రభావితమైన            JJ      <fs af='ప్రభావితమైన,adj,,,,any,,'>
2.2     ఔలీ             NNP     <fs af='ఔలీ,n,n,sg,3,,0,0' name=Oli poslcat=NM>
```

	))		
3	((	NP	<fs af='భారత్,n,n,sg,3,d,యొక్క,0' head=BArawa drel=r6:4 name=3>
3.1	భారత్యొక్క	NNP	<fs af='భారత్,n,n,sg,3,d,యొక్క,0' name=BArawa>
	))		
4	((	NP	<fs af='పర్యాటకుడు,n,m,sg,3,d,0,0' head=paryataka name=4>
4.1	ప్రసిద్ధ	JJ	<fs af='ప్రసిద్ధ,adj,,,,any,,'>
4.2	పర్యాటక	NN	<fs af='@-పర్యాటక,n,n,sg,3,o,0,0' name=paryataka>
	))		
5	((	NP	<fs af='స్థలం,n,n,sg,3,d,0,0' head=sWala name=5>
5.1	స్థలం	PRP	<fs af='స్థలం,n,n,sg,3,d,యొక్క,0' name=sWala>
	))		
6	((	VGF	<fs af='ఉండు,v,n,sg,3,,ఆ,hE' head= drel=head=hE name=6>
6.1	.	NN	<fs af='.,punc,,,,,,' poslcat=NM>
	))		

4. Conclusion:

The architecture of this system is based on analyze-transfer-generate paradigm. The flow of the input sentence in the system as shown in fig:1. All the modules have been integrated on the dashboard, a tool, where the data flow in the pipeline is configured. This ensures speed, since it uses shared memory. The MT system demonstrated here is a completely automated translation system without involving human intervention at any stage from analysis, processing and generation. Though the current system is built for the tourism domain, it can be extended to any other domain. The system can be used to translate web pages or text material from books, magazines, newspapers etc. written in standard language. Though the output of MT system will never be as human translated text, yet it is evaluated (Mona et.al) against certain criteria specifying thresholds of understandability, comprehensibility and accuracy. This project sets its goal, to produce output comprehensible and for this, the evaluation reference set of one thousand sample input sentences (with syntactic structures having nesting up to two clauses deep), with their expected output, manually translated, has been provided with.

Sampark runs on Linux platform with Apache-2.0 server. The browser used for the online translation can be Firefox 1.0.4, IE 6.0 or Mozilla 1.7.8.

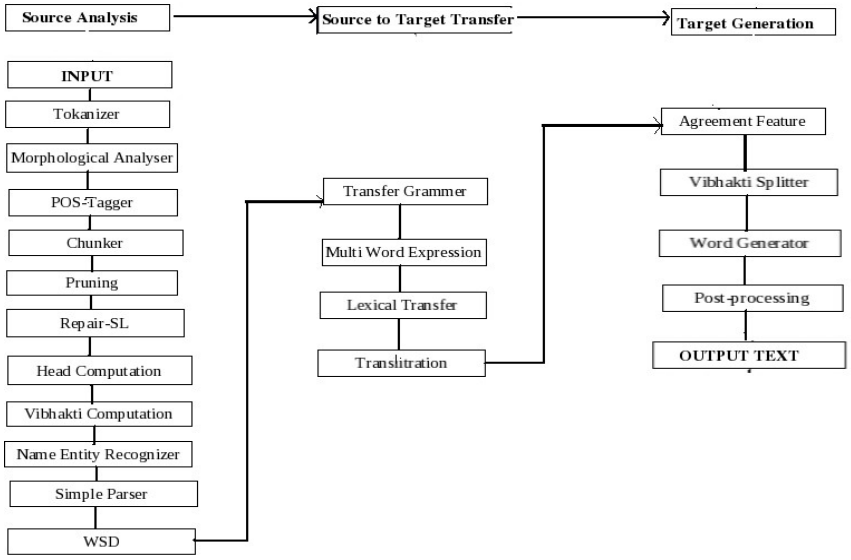

Fig:1

References:

Akshar Bharati, Rajeev Sangal and Dipti Mishra Sharma. 2009. SSF: Shakti Standard format. Report No: IIIT/TR/2009/85. CLTRC, IIIT-Hyderabad

Akshar Bharati, Mridual Gupta, Vineet Yadav and Dipti Mishra Sharma. 2009. Simple Parser for Indian Languages in Dependency Framework. In 3rd Linguistic Annotation Workshop (LAWIII), SIGANN, 47th ACL- 4th IJCNLP, Singapore.

Uma Maheswar Rao G. and Christopher M. 2010. Word Synthesizer Engine. In Morphological Analyzer and Generators. Mona Parakh (ed.) Page 73-81. Mysore; CIIL.

Uma Maheswar Rao G. and Parameshwari K. 2010. On the Description of Morphological Data for Morphological Analysers and Generators: A case of Telugu, Tamil and Kannada. In Morphological Analyzer and Generators. Mona Parakh (ed.) Page 114-123. Mysore; CIIL.

ILMT Consortium. 2007. *ILMT SRS and Functional Specifications (mimeo)*. Hyderabad.

Uma Maheswar Rao G, Amba P. Kulkarni and Christopher M. 2007. *Functional Specifications of Morphology* (memo). Hyderabad.

Anthes G. 2010. Automated Translation of Indian Languages. ACM 53(1).

EAGLES. 1996. *Expert Advisory Group of Language Engineering Evaluation of Natural Language Processing system (Final Report). DG XII of European commission.*

Moona R. Singh, Sangal R, Sharma D. 2004. M. *MTeval: A Evaluation methodolgy for Machine Translation system.* In SIMPLE'04 (p 15-19) at IIT-Kharagpur 19 – 21 March, 2004.

Sangal R. 2005. *Dashboard: A FRAMEWORK FOR NATURAL LANGUAGE PROCESSING*. http://www.sumobrain.com/patents/wipo/Framework-natural-language-processing/WO2009087431.html

Pawan Kumar, Rashid Ahmed, Rathaur A.K, Mukul K. Sinha and Sangal R. 2010. *Re-Engineering Machine Translation System Through Symbiotic Approach.* Spingerlink (p 193-204). Third International Conference on Contemporary Computing IC3, August, Nodia, India.

[i]These bi-directional Indian Language to Indian Language Machine Translation Systems (IL-ILMT) are planned to be developed by a consortium of IL-ILMT constituted (and funded) by Govt. of India.