Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6 – No.8, March 2017 – www.caeaccess.org

Morphological Analysis of Manipuri Language

Yumnam Bablu
Assam University
Computer Science

Ch. Yashawanta
Manipur University
Linguistics Department

Nameirakpam Amit
Manipur University
Linguistics Department

ABSTRACT

Morphological analysis forms the basic foundation in NLP applications including syntax parsing Machine Translation (MT), Information Retrieval (IR) and automatic indexing in all languages. It is the field of the linguistics; it can provide valuable information for computer based linguistics task such as lemmatization and studies of internal structure of the words. Computational Morphology is the application of morphological rules in the field of computational linguistics, and it is the emerging area in AI, which studies the structure of words, are formed by combining smaller units of linguistics information, called morphemes: the building blocks of words. Morphological analysis provides about semantic and syntactic role in a sentence. It analyzes the Manipuri word forms and produces several grammatical information associated with the words. The Morphological Analyzer for Manipuri has been tested on 3500 Manipuri words in Shakti Standard format (SSF) using Meitei Mayek as source; thereby an accuracy of 80% has been obtained on a manual check.

General Terms

Machine Translation, NLP, Morphological Analysis.

Keywords

Manipuri; Meitei Mayek; Computational Morphology, Information Retrieval, SSF

1. INTRODUCTION

The first step in natural language processing is to identify words in a sentence. The process is called morphological analysis. The Manipuri Morphological Analyzer is built using the methodology of finite-state compilers and algorithms, and the results are stored and run as finite-state transducers. Manipuri language also known as Meiteilon or Meiteiron belongs to the Kuki-Chin [1] branch of the Tibeto Burman language, sub-family of Sino Tibetan Language. Manipuri language is an official language as well as a Lingua franca among the various speech communities [2]. Manipuri has been adopted as the medium of instruction and examination from the primary to the high school stage. It has been the state language of Manipur since the 8thcentury A.D. Manipuri language has been recognized as the 8thscheduled language in the Indian Constitution since 1992[3]. Short Taxonomy of Manipuri word formation

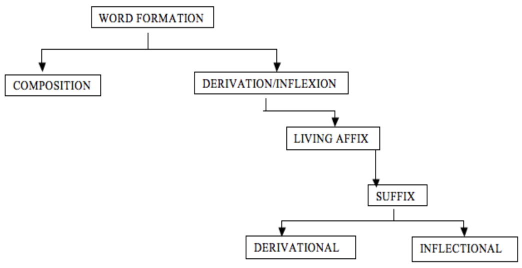

Figure 1: Word Formation in Manipuri.

Morphology consists of two branches: inflectional morphology and derivational morphology. Inflectional morphology is the study of those processes of word formation where various inflectional forms are formed from the existing stems [4].

Examples1:
Plurals: boroI–>boroIsiQ
Aspect: chai –>chaari,

Derivational morphology is the study of those processes of the word formation where new words are formed from the existing stems through the addition of morphemes. The meaning of the resultant new word is different from the original word and it often belongs to a different syntactic category [4]. Examples:

- Adjective to verb: acAb –>cAb
- Adjective to adjective: acAb –>acAbgI
- Noun to adjective: cArIb –>acAb

2. PROBLEM STATEMENT

About the problem statement we will list four main objective questions as follows:

a. What are morphological categories in Manipuri Language?

b. What are computational morphological process in agglutinative language like Manipuri?

c. What are rules involved in morphological process?

d. What is the computational model for Morphological analysis in Manipuri Language?

e. What is the best approach for computational Morphological analysis?

3. OBJECTIVES OF MAM ANALYSIS

Manipuri being agglutinative language is highly inflectional language, which have the capability of generating more then thousands of words from a single root. Hence morphological analysis is vital for high-level applications to understand various words or lexeme in our language. So morphological analysis of Manipuri language forms the foundation for applications related with NLP. Agglutinative languages show high morpheme per word or head word ratio and have complex morphotactics structures, the absence of fusion at affix boundaries make the task of segmentation fluent once the architecture or the model for implementation of morphotactics is build. Here we are listing our main objectives of MAM analysis:

a. To identify the morphological categories of Manipuri Language.

b. To identify the number of prefix, Suffix as per category wise.

c. To identify the morphological categories in Manipuri Language.

d. To identify the computational process in agglutinative language, Manipuri.

e. To identify the rules of MAM in Manipuri Language.

f. To identify the best approached for MAM.

4. METHODOLOGY

The process goes in four modules i.e.

1. Data collection.
2. Classification.
3. Analysis.
4. Implementation.

As a data collection process we start making paradigm table in category wise of word classification.

a. Lexical category (lcat) has all these possible values like:

1.	Noun (singular/plural)	= n
2.	Verb	= v
3.	Adjective	= adj
4.	Adverb	= adv
5.	Pronoun	= pn
6.	Nlocative	= nst
7.	Avya	= avy
8.	Postposition	= psp
9.	Number	= num
10.	Ordinal	=ord
11.	Punctuation	=punc
12.	Unknown	=unkn

b. The possible values for Gender: m, f, n , mf , mn , fn, any .

c. The possibles values for Number: sg, pl, dual, any.

d. The possible values for Preson : 1, 2, 3, any.

e. The possible values for Case : d(direct),o(oblique)

f. The possible case marker: dir, obl, ki , ku, ni , nu, lo, wo, yoVkkaetc ...

g. The possible feature structre for the word which is unknown to morph is <fsaf='word,unkn,,,,,,'>

h. The possible feature structre for the punctation mark which is unknown to morph is <fsaf='&sdot;,punc,,,,,,'>for \. .

i. The possible feature structure of the number is <fsaf='88,num,,,,,,'>

j. The possible values for case name: ex: nom, acc, dubi, etc **or** 1, 2, 3

This file is read by morph in compiler mode during paradigm-data input.

5. MORPHOLOGICAL ANALYZER

Morphological analyzers perform morphological analysis. There are some important approaches for executing morphological analysis. But the two approaches, which are used widely, are:

a. Finite state machines based approach

b. Machine learning approach

a. Finite State Machines Based Approach

Section describes the Finite state machines approach used for building Finite State Transducers (FST) based morphological analyzers.

Resources

The main goal of this approach is to list all the possible parses/analyses of an input word. In order to build a morphological parser using an FST based approach, the following resources are used in general:

1. Lexicons

Lexicon of a language is its vocabulary or the list of all words in Manipuri or particular languages. It is an explicit list of every word of the language. It is cumbersome to list every word in a language. Hence generally computational lexicons are used for this purpose. The Finite-state automaton (FSA) is generally used to model lexicons. A structured collection of the entire morpheme i.e. the root or headwords and morphemes or affixes of the words are collected [5].

Table 1. Lexicon Table

Sl.no.	Lexicon	Numbers
1	Headwords	13463
2	Noun Suffix	37
3	Pronoun Suffix	39
4	Numerals	32
5	Verb Suffix	100
6	Adverb Suffix	23
7	Adjective Suffix	31
8	Prefix	9

2. Morphotactics

This explains the morpheme ordering ex: the plural morpheme follows the main noun morpheme. Example: *cars* = *car* + N + pl. 3) Orthographic rules: these rules are also known as spelling rules. They model changes when two morphemes combine. Example: fox + plural s = foxes. Here the *e*-insertion rule is applied.

The number of paradigms per PARADIGM-INPUT-FILE is limited to one. More than one paradigm definitions in the same file will lead to the particular file being rejected by morph for having improper no of word forms.

Table 2. Paradigm Table of Noun

əhum (root)	**Category**	**Suffix**
əhum	num+nom	-n
əhum-n-di	num+nom+spec	-n-di
-	-	-
-	-	-
əhum-du-gi-di	num+det+ben+spec	-du-gi-di

Table 3. Paradigm Table of Pronoun

mədu (root)	**Category**	**Suffix**
mədu-n	pro-pl+nom	-n
mədu-gi-di	Pro-pl+Gen+Spec	-gi-di
-	-	-
-	-	-
mədu-si-n-di	Pro-pl+det+Conts+Spec	-si-n-di

TABLE 4. PARADIGM TABLE OF VERB

***KNn* (root)**	**Category**	**Suffix**
KNn-ri	verb+dur	-ri
KNn -gni	verb+future	-gni
-	-	-
-	-	-
KNn -gL-li	verb+ habitual+asp	-gL-li

TABLE 5. PARADIGM TABLE OF ADJECTIVE

yamdrbə (root)	**Category**	**Suffix**
yamdrbə-si-n	adj+det+nom	-si-n
yamdrbə-si-n-di	adj+Det+nom+spec	-si-n-di
-	-	-
-	-	-
yamdrbə-du-bu-n	adj+dt+acc+conts	-du-bu-n

TABLE 6. PARADIGM TABLE OF NUMBER

əmə (root)	**Category**	**Suffix**
əmə-si-n	num+det+nom	-si-n
əmə-du-n	num+Det+nom	-du-n
-	-	-
-	-	-
əmə-du-n-di	num+det+conts+spec	-du-n-di

Table 7. Paradigm Table of ordinal

məŋasubə(root)	**Category**	**Suffix**
məŋasubə-si-n	num+Det+nom	-du-n
məŋasubə-si-n-di	num+det+nom+spec	-si-n-di
-	-	-
-	-	-
məŋasubə-si-dgi-n	num+det+abl+conts	-si-dgi-n

Paradigm table are used to morphological database. These data are useful as a linguistics source to improve the Manipuri language. Morphological Analysis using Finite State Transducers in Manipuri language is given below as FST for lexicon formation of Manipuri words from one root.

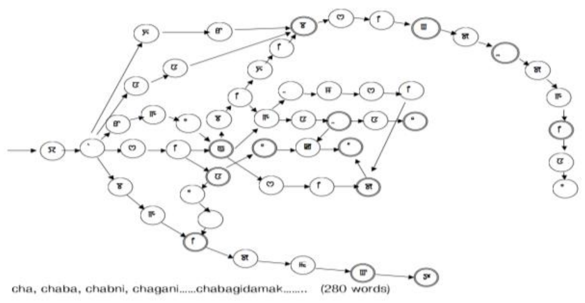
cha, chaba, chabni, chagani......chabagidamak........ (280 words)

Figure 2: Formation of Lexicon from one root in more then 280 (might be more) words

Flow Chart of Morph will show the overview of the program control from one module to the other. It shows the decision points in the program

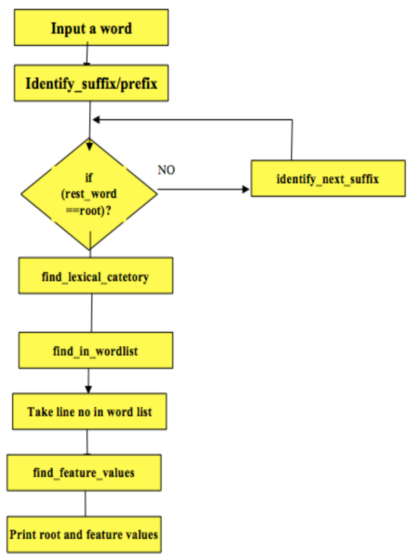

Figure 3: Flow Chart for MAM

DFD diagram showing how data are flow in MAM.

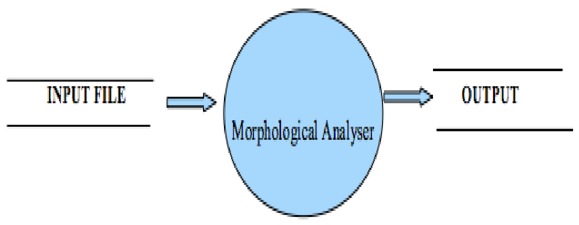

Figure 4: Zero Level DFD.

Data flow in another way:

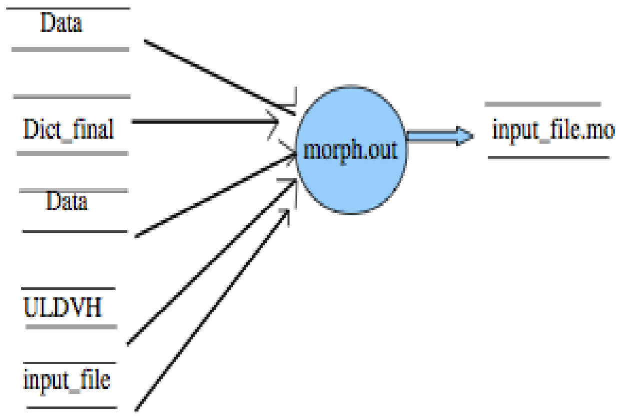

Figure 5: Zero Level DFD

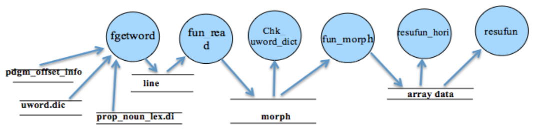

Figure 6: First Level DFD

Here the Figure 7 shown below DFDs is represented as per the programming concept. How data flows was shown from LEVEL 0 to 3

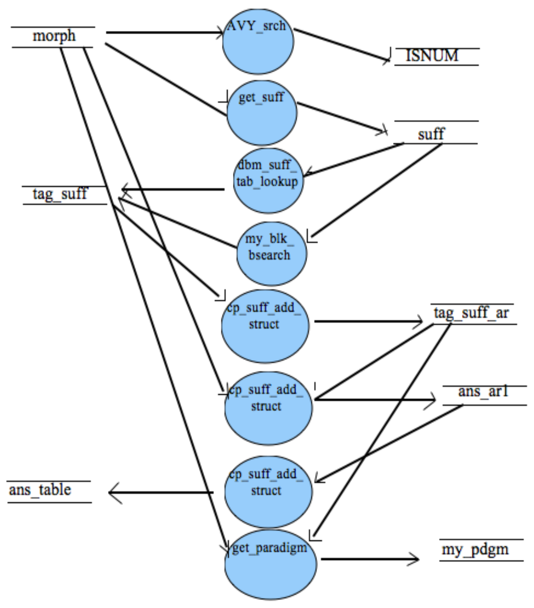

Figure 7: Second Level DFD

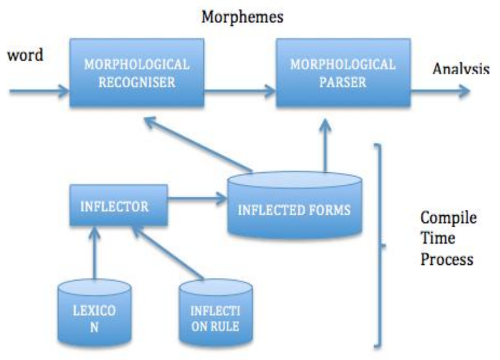

Figure 8: Architecture of MAM

6. MORPHOLOGICAL ANALYSIS

In Morphological analysis we take words and we try to identify the suffix or prefix, and check weather this suffix is a valid suffix. If this suffix is present in the word-list then it is a vialed if not it's an unknown word for Morphological Analyzer. If the suffix is valid then check weather the stem is valid or not just, by converting it to root word, by adding and deleting is done. If the suffix and root words are valid then take the line number of the word in word-list then get the feature structure (like gnp, tam, case, case marker value). Add root word and feature structure to API- wrapper to print in the data tree.

```
# This program is use as pre-processing module before tokenizer
binmode(STDIN, ":utf8");
binmode(STDOUT, ":utf8");
while($line=<>)
{
        utf8::decode($line);
$line =~ s/\x{2018}/'/g; # <2018> ‘ is Replaced by single quote "'"
$line=~s/\x{2019}/'/g; # <2019> ’ is Replaced by single quote "'"
$line=~s/\x{201C}/"/g; # <201C> “ is Replaced by single quote "
$line=~s/\x{201D}/"/g; # <201D> ” is Replaced by single quote "
$line=~s/\x{200D}//g; # <200D> is Removed
$line=~s/\x{200C}//g; # <200C> is Removed
$line=~s/\x{feff}//g; # <feff> is Removed
$line=~s/\x{0D}//g; #
is Removed
        print $line;
}
```

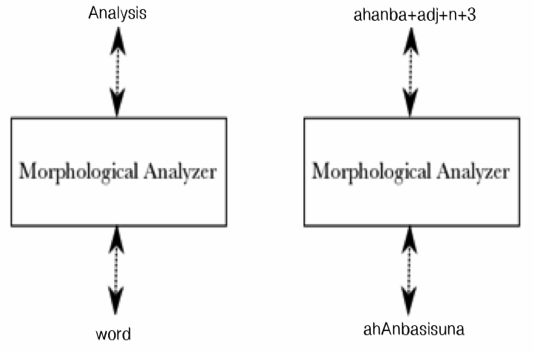

Figure 9: Process of MAM

A Manipuri Morphology compiler is a program, which compiles and analyses words belonging to a Manipuri language (Meiteilon). It works in Manipuri language using Meitei Mayek; hence it can learn and recognize words from Manipuri language, which are written in Meitei script. Morph has two modes of operation, via the COMPILER mode and the ANALYSER mode. In the COMPILER mode it reads information about the words of a language from paradigm-input files and lexicon-data files and stores the processed information. This process is referred to as "compilation of data". Once this has been done Morph can recognize the words, which were present in the data. All the data need not be compiled in one go, fresh data can be fed to morph any time by running it in compiler mode. To recognize words one has to run morph in its second mode of operation, which is the ANALYSER mode.

It will recognize only those Manipuri words, which it has been "taught"i.e. trained data. It outputs all the descriptions of the given word it has read during compiler mode. It produces a diagnostic "Unknown word <given word>" to say that it does not find the word in its list. Compilation Manipuri data is done by running morph in compiler mode. While "compiling morph" means putting the morph source code through the C-compiler, which may be necessary especially to customize morph program to some specific need. Compilation of morph is not connected with "data compilation" or "compilation of data" in any manner. At times changes may demand recompilation of full data, while for some changes only recompilation of morph may suffice, and for some changes both may be called for.

Input- output Specifications:

Input:	TKN
Output:	Morphological Analysis

Input specifications
It require that property TKN_ must be defined in the input SSF that is given to the Morphological Analysis

ADD	TKN	CAT
1	rAmA	<UNDEF>
2	sIwA	<UNDEF>

Output specifications: required that Morphological Analyzer as given below would define property of attribute feature. So the output SSF must contain Feature structures to all the valid values.

Input specifications require that property TKN_ must be defined in the input SSF that is given to the Morphological Analyzer.

Output specifications required that Morphological Analyzer as given below would define property of attribute feature. So the output SSF must contain Feature structures to all the valid values.

Output:

An output SSF from Morph must contain all the **four** columns of SSF.
The first column will have ADDR
The second column will have TKN
The third column will have CAT as UNDEF
The fourth column will have the feature structure, *fs*
The feature structure will be in the form of either abbreviated features, of and/or attribute-value pairs. If it has two feature structures then separate them with **" | "** character and between each column there is a tab that separates the fields.

The possible values of fs is listed below,

NOUNS:
A noun is analysed as root+suff+{features(such as gender, number,...)}.

The complete structure is presented below.

<fs af root = "Root of the word", lcat = "Lexical category of the root", gend ="Gender of the word", num = "Number coressponding to the word form", pers = "Person of the word", case = "Case (Direct / Oblique)", vibh = "(cm / tam)" case_name= "case name", spec= "Specificity Marker", emph ="Emphatic Marker", dubi = "Dubitative Marker", interj = "Interjection Marker" conj = "Conjunction Marker" hon ="Honorific Marker" agr_gen ="Gender of the agreeing noun" agr_num ="Number of the agreeing noun" agr_per ="Person of the agreeing noun" suff ="Form of suffix representing all the above markers">

VERBS:
The verb analysis structure is presented below.
<fsaf root = "Root of the word", lcat = "Lexical category of the root", tam ="Suffix indicating Tense Aspect Modality", gend = "Gender of the word", num = "Number corresponding to the word form", pers = "Person of the word", spec = "Specificity Marker", emph = "Emphatic Marker", dubi = "Dubitative Marker", interj = "Interjection Marker", conj = "Conjunction Marker"hon = "Honorific Marker", neg = "verb-neg Marker", voice = "Voice", caus = "Whether the verb form is causative or not(y/n)" finiteness = "Whether the verb form is finite or not (y/n)", suff = "Suff representing all the above markers">

ADJECTIVES:
The feature structure for Adjectives is as follows:
<fsaf root = "Root",lcat = "Lexical category",gend = "Gender of the word",num = "Number",pers = "Person of the word", degree= "degree",-like = "like", dubi = "Dubitative",interr = "Interrogative", emph = "Emphatic", conj = "Conjunction Marker", ?spec = "Specific", suff = "complete suffix">

ADVERBS:
The feature structuer for Adverbs is presented below.
<fs af root= "Root of the word", lcat= "Lexical category of the root", dubi= "Dubitative Marker", interr= "Interrogative", emph= "Emphatic Marker" conj= "Conjunction Marker", ?spec= "Specific", suff= "complete suffix">

NOUN LOCATIVE:
The complete structure is presented below.
<fs af root = "Root of the word", lcat = "Lexical category of the root", gend ="Gender of the word", num = "Number coressponding to the word form", pers = "Person of the word", case = "Case (Direct / Oblique)", vibh = "(cm / tam)", spec = "Specificity Marker", emph = "Emphatic Marker", dubi = "Dubitative Marker", nterj = "Interjection Marker", conj = "Conjunction Marker", case_name= "case name", hon = "Honorific Marker", agr_gen = "Gender of the agreeing noun", agr_num = "Number of the agreeing noun", agr_per = "Person of the agreeing noun", suff = "Form of suffix representing all the above markers">

NUMBER :

A Numeral is analysed as root+suff+{features(such as gender, number,...)}.

The complete structure is presented below.

<fs af root = "numer", lcat ="num" gend ="", num ="", pers ="", case ="", vibh ="", suff ="" >

PUNCTUATION :
A Punctuation is not analysed just give the NCR.

The complete structure is presented below.

<fs af root = "NCR" lcat ="punc" gend ="" num ="" pers ="" case ="" vibh ="" suff ="" >

UNKNOWN :
A Unknown is not analysed and it is just repeatd.

The complete structure is presented below.

<fs af root = "word", lcat ="unk", gend ="", num ="", pers ="" case ="", vibh ="", suff ="">

7. IMPLEMENTATION

The code is separated to into parts (subroutines) like – command line parsing, program initialization, error handling and logging, and the main application. All the function/subroutines have been defined in the defn.h, struct.h struct1.h files. It would be better if,

- Functions/subroutines are defined in the separate program files, .cpp file.

- In the main program they must be called as subroutines

When a new paradigm file is added, or in an existing .p file # of lines is changed:

a. p file should be placed in pc_data sub dir.

b. Relevant info regarding the category, features & its values should be entered in Ca, Ce & Fe files in test area.Program to convert Meitei Mayek to WX mapping.

The given Perl programs will convert Meitei layout to wx mapping.

```
while ($line=<>){
        $line=~s/k([^AeiouO])/ka$1/g;
        $line=~s/s([^AeiouO])/sa$1/g;
        $line=~s/l([^AeiouO])/la$1/g;
        $line=~s/m([^AeiouO])/ma$1/g;
        $line=~s/p([^AeiouO])/pa$1/g;
        $line=~s/n([^AeiouO])/na$1/g;
        $line=~s/c([^AeiouO])/ca$1/g;
        $line=~s/t([^AeiouO])/ta$1/g;
        $line=~s/K([^AeiouO])/Ka$1/g;
        $line=~s/q([^AeiouO])/fa$1/g;
        $line=~s/q/f/g;
        $line=~s/T([^AeiouO])/wa$1/g;
        $line=~s/T/w/g;
        $line=~s/w([^AeiouO])/va$1/g;
        $line=~s/w/v/g;
        $line=~s/y([^AeiouO])/ya$1/g;
        $line=~s/h([^AeiouO])/ha$1/g;
        $line=~s/P([^AeiouO])/Pa$1/g;
        #$line=~s/a([^AeiouO])/aa$1/g;
        $line=~s/g([^AeiouO])/ga$1/g;
        $line=~s/J([^AeiouO])/Ja$1/g;
        $line=~s/r([^AeiouO])/ra$1/g;
        $line=~s/b([^AeiouO])/ba$1/g;
        $line=~s/j([^AeiouO])/ja$1/g;
        $line=~s/G([^AeiouO])/Ga$1/g;
        $line=~s/D([^AeiouO])/Xa$1/g;
```

8. CONCLUSION & FUTURE WORKS

In the present work, the development of a Manipuri Morphological Analysis has been described. The root dictionary stores the related information of the corresponding roots. The Analyzer can classify the word classes and sentence types based on the affix information. The verbs are under bound category. The verb morphology is more complex than those others. The distinction between the noun class and verb classes is relatively clear; the distinction between nouns and adjectives is often vague. Thus, the assumption made for word categories depend upon the root category and affix information. In the stripping of the morphemes the various morphemes pattern combinations are tested.

The Natural Language Processing tools need more text corpus with better transfer rules and techniques to achieve quality output. The performance of the various Manipuri NLP tools that have been developed in the present work need to be improved by experimenting with various machine-learning approaches with more training data. Future works include the developments of automatic Morphological Analyzer using some machine learning algorithms. The exploration and identification of additional linguistics factors that can be incorporated into the Morphological Analysis to improve the performance is an important future task.

9. ACKNOWLEDGMENTS

We thanks to Shallow Parser Tools of Indian Language (SPTIL), DIT, Project Consortium Leader Prof. G. Uma Maheswar Rao and the teams. And Indian Language Corpora Initiative (ILCI), DIT project PI, Girish Nath Jha.

10. REFERENCES

[1] Grieson, 1973. Linguistics Survey of India,Vol III part III.

[2] Ch.Yashawanta Singh. 2000. Manipuri Grammar. Rajesh Publications, New Delhi.

[3] Sarangi, A. (2009). Language and Politics in India. (P: 27). New Delhi, Oxford University Press

[4] Nikhil, K., Abhilash, I. and Sharma, D. M. Hindi Derivational Morphological Analyzer. In Proceedings of SIGMORPHON, 2012.

[5] Web enabled Multilingual Manipuri Dictionary, Yumnam Bablu Singh.

[6] Aksher Bharathi, Rajeev Sangal, et.al, "Natural Language Processing: A Paninian Perspective".