CALTS Morphological Generator:
A Reverser Engineering Approach using Morphological Analyzer Database

Christopher[1], M. Uma Maheshwar Rao[2], G.

[2]CALTS, University of Hyderabad,
[1]LTRC, International Institute of Information Technology
Hyderabad-46.
[2]efthachris@gmail.com, [1]guraohyd@gmail.com

Abstract:

This paper describes the development of a Morphological Generator, a generator that is built with the reverse engineering approach using Morphological Analyzer database. It demonstrates that the data base that is used for Morphological Analyzer (MA) can be used for word generation too, which can be called as reverse engineering Approach. This Generator synthesizes all and only the well-formed word forms. These word forms include both inflectional and derivational forms. This Morphological Generator engine is independent of language and works effectively and is based on word-and-paradigm method. This Computational model uses machine learning method based on morphological data base developed using word and paradigm model of Morphology. The database is taken from CALTS Morphological Analyzer. This method not only ensures coverage but also evolvement.

The engine takes input a root and along with it its inflectional categories (features) like gender, number, person and case in case of nouns and verbal categories in case of verbs and other relevant inflectional endings depending on the category.

In this paper we describe how the Morphological Generator handles all of the inflectional forms in addition to the productive derivational forms. When tested with languages like Telugu, Hindi and Tamil their accuracy was 97.2%, 98% and 94% respectively.

1. Introduction:

One of the crucial integral parts of the major Natural Language Processing (NLP) applications is morphological generator or word synthesizer. Morphological analyzer and generator are two essential and basic tools for building a system like a machine translation. A Morphological analyzer processes a word and analyses it into its root, along with its grammatical information depending upon its word class. Morphological generator does exactly the reverse of it, i.e. given a root and the relevant morphological category and the grammatical information it generates the word form of that root, a projection of its morphological category. *When the task of a Generator is same as reverse of the Analyzer, it is enough to reverse the process using the same linguistic resources.* Having this basic logic the the present Morphological Generator is built. This Morphological Generator is based on use of the roots and the paradigm information from the CALTS Morphological Analyzer's lexicon, The same set of feature-values and rules of add and/or delete rules, which are generated when the paradigmatic database of analyzer is compiled. The rest of the paper gives a detailed description of the Morphological Generator and its working. The Morphological Generator will be called as WordGen hereafter in this paper.

2. Organization of Morphological Data :

The CALTS Morphological Analyzer analyzes all the inflectional as well as productive derivational word forms. It is based on word-and-paradigm method. The Word-and-Paradigm assumed here involves an exhaustive collection of each and all wordforms relatable to a lexeme, collectively called as *Paradigm*. This requires the identification of all the inflectional categories in the language, identification of conjugational and declensional classes of paradigms and finally listing of all paradigmatic members for each of these. The definitions of each of the Paradigmatic form in the Paradigm list are given in *Feature Value table*.

A *Root word dictionary* i.e. the lexicon type in the Morphological Analyzer differs from a conventional dictionary**.** The dictionary for Morphological Analysis which is built for Word and Paradigm Model contains roots, categories and their corresponding paradigm. The Present Morphological analyzer lexicon contains root/lemma, i.e. the part of the lemma which is common to all the inflected forms, and the paradigm name. Compiling involving wordforms present in the *Paradigm*, root words from the *Root word dictionary* and Feature Values genetare a set of add/deletion rules (may be called as *morphophonemic Rules)*.

The Morphological Generator, the WordGen, is built on the basis of the morphological rules extracted from the compilation of the relevant and exhaustive listing of paradigmatic forms. These sets of paradigmatic forms with a shared lexeme describe the morphology of the language. The lexeme is matched against each wordform for a common maximally matched sequence and extracting the unmatched portion as formative/functional element which stands for the feature value.

For example: Consider the following members of a paradigm.

Root/Lexeme: ***winu,v.***

Paradigmatic Members	Lexeme	Common Maximal match	Functional/ Formative Element
winnAdu	Past-m-sg-3	*win*	nAdu
wiMtAdu	np-m-sg-3	*wi*	MtAdu
winadu	neg-m-sg-3	*win*	adu
wini	nf-past	*win*	i
wine	nf-adjl-ppl	*win*	e
winu	imp-sg	*winu*	0

The generator accepts roots and their morphological information in terms of category and the functional elements to generate all the corresponding forms. The Morphological Generator uses the compiled resources of the morphological analyzer data base to generate word forms. It uses root word dictionary, Feature value Table and morphophonemic rules, as described below:

2.1) The Lexicon:

The lexicon is a dictionary containing a list of roots/lexeme, each with its lexical category and paradigm type. It is organized in the form of a simple linear, non-hierarchical, sequence of the root delimiter (**,**) lexical category delimiter (**,**) and the paradigm type.

S.No	Root/Lexeme	Lexical Category (lcat)	Paradigm Type
1	*winu*	v	*koVnu*
2	*maMwri*	n	*gaxi*
4	*welika*	adj	*lewa*
5	*appudu*	adv	*appudu*
6	*iwadu*	pn	*vAdu*

Table-1: Lexicon

2.1) Feature value Table:

The Feature Value Table is essentially list of affixes with their morpho-syntactic feature values like *gender*, *number*, *person* and the relevant morphological category information stored in the form of a table. **Feature value Table** contains lcat, affix and case associated with nouns, pronouns and tense, aspect, modal categories with or without gender, number and person associated with verbs, the inflectional affixes associated with Adjectives and locative nouns.

S.No	Rule No.	lcat	Affix	Gender	Number	Person
1	719	v	*iwi*	m	pl	2
2	653	n	*wopAtu*	null	sg	null
3	1649	pn	*lekuMdA*	null	pl	null
4	963	adj	*ti*	null	pl	null

Table 2: Feature Value Table

2.3) Synthesis Rule Set (morphophonemic Rules):

Maximally projected wordforms are generated by an exhaustive set of concatenation rules. The synthesis rule set is an exhaustive rule set, essentially a combination of concatenation processes which add the desired suffixes to the given root/lexeme and which itself is appropriately modified by the relevant deletion rules. Both the add rule and deletion rule may apply vacuously in case the value of the character string to be added or deleted is null.

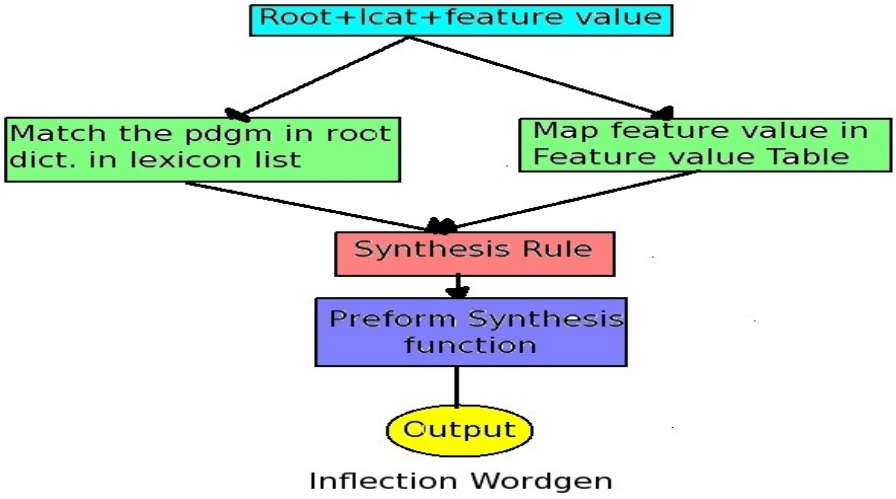
S.No	Add-Affix	Del-base/root	Paradigm name	Rule No.
1	*A*	*u*	*vAdu*	1649
2	*IsAdA*	*iyyi*	*wiyyi*	653
3	*akuMdA*	*u*	*poVg?du*	719

4	*NNilekuMdA*	*du*	*snehiwudu*	963

Table 3 : Synthesizer Rule Set

3. Computational Model of WordGen

System Architecture

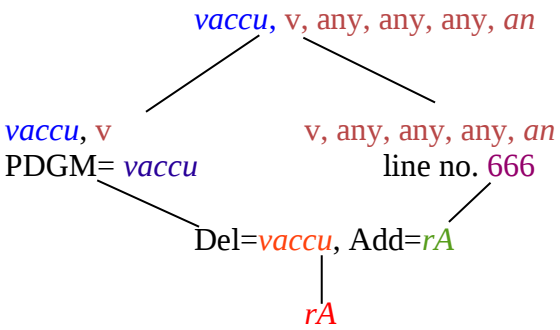


Inflection Wordgen

The above given picture is the model of *WordGen*. The architecture here involves the synthesis of word forms starting from the given root and the desired features, finding its category and the paradigm type in the lexical database, then search carried out for the line in the synthesis table where the set of morpho-syntactic feature values are listed. Then accordingly carry out delete and add functions, which involve the modification of the given root by the selection of the appropriate allomorph from the add rule followed by concatenation in the synthetic rule set.

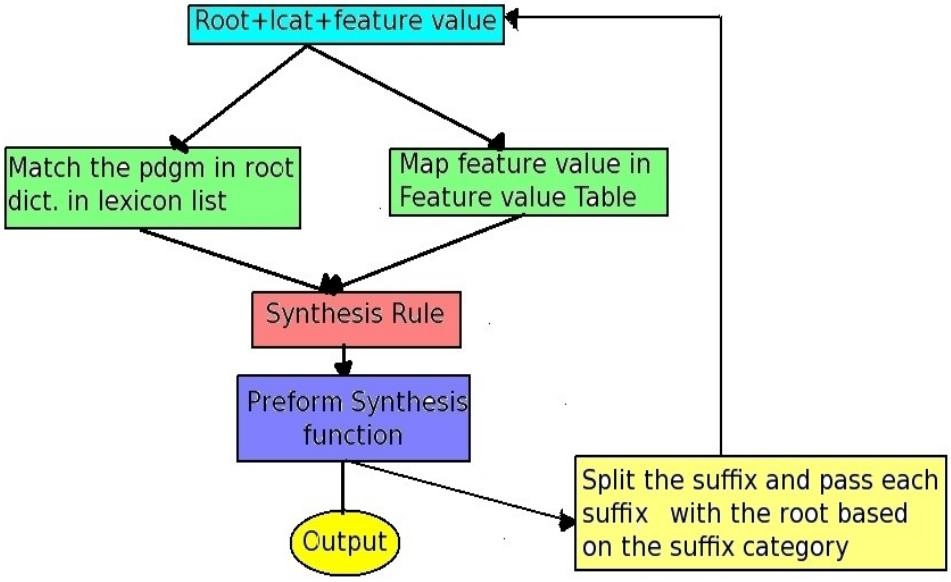
The working of the *WordGen* can be viewed in step by step process, by using the data resources.

Root word= *vaccu*, lexical category=v, gender= any, number=any, person=any and suffix=*an*



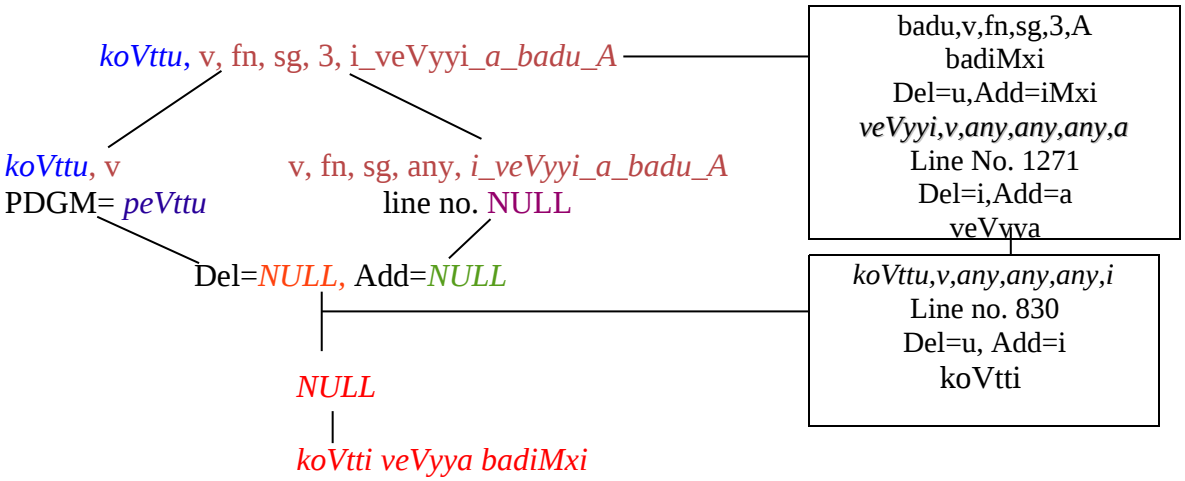
It is able to generate the wordforms with inflectional morphology, but inorder to generate some productive derivational morphological forms a new technique has been introduced. A Floating Lexicon is devised to include derivational or compounding components of words. The

Basic Architecture for this type of derivational module of WordGen is given below.



Derivation Wordgen

The working of the Productive *WordGen* can be viewed in step by step process, by using the following data resources.
Root word = *koVttu*, lexical category = v, gender = fn, number = sg, person = 3 and suffix = *i_veVyyi_a_badu_A*



4. Input and Output Sepcification:

Input for this computational model of Morphological Generator is in S*haskthi Standard Format (SSF)*. Where we have a token number, token, pos-tag and its morphological analysis. All these are in different fields (Coloumns). It reads the fourth coloum's i.e *morph analysis*, in which *1st field is root, 2nd is lex.cat, 3rd is gen, 4th is num, 5th is per, 6th is case (d/o), 7th is case*

marker/tam, 8th is suffix. By using all the seven elements of the Morphological Analysis, Generator generates the wordforms and modifies the 2nd column i.e *token* of the SSF format.

Input in SSF:

```
<Sentence id="1">
1       ((      NP      <fs af='rAmudu,n,m,sg,3,d,ku,ku'>
1.1     rAmudu          NN      <fs af='rAmudu,n,m,sg,3,d,ku,ku'>
        ))
2       ((      NP      <fs af='Akali,n,m,sg,3,d,0,ku'>
2.1     Akali   NN      <fs af='Akali,n,m,sg,3,d,0,ku'>
        ))
3       ((      VGF     <fs af='veVyyi,v,m,sg,1,,A,A'>
3.1     veyyi   VM      <fs af='veVyyi,v,m,sg,1,,A,A'>
        ))
</Sentence>
```

Output in SSF:

```
<Sentence id="1">
1       ((      NP      <fs af='rAmudu,n,m,sg,3,d,ku,ku'>
1.1     rAmudiki        NN      <fs af='rAmudu,n,m,sg,3,d,ku,ku'>
        ))
2       ((      NP      <fs af='Akali,n,m,sg,3,d,0,ku'>
2.1     Akali   NN      <fs af='Akali,n,m,sg,3,d,0,ku'>
        ))
3       ((      VGF     <fs af='veVyyi,v,n,sg,1,,A,A'>
3.1     vesiMxi         VM      <fs af='veVyyi,v,n,sg,1,,A,A'>
        ))
</Sentence>
```

6. Conclusion and Results

WordGen generates wordforms for all the lexical classes: nouns, pronouns, verbs, adjectives, locative nouns and number words. This generator is first of its kind which can handle inflectional and productive derivational morphologies. Which shares it database with Morphological Analyzer. If Analyzers coverage increase, accuracy of the Generator also goes up. Current version of the tool is integrated with IL-ILMT Hindi-Telugu, Telugu-Hindi, Telugu - Tamil and Tamil-Telugu systems (CALTS, UoH).

REFERENCES

1. Krishnamurti, Bh. 1985. A Grammar of Modern Telugu. Delhi: Oxford University Press.

2. Uma Maheshwar Rao, G. 1999. A Morphological Analyzer for Telugu (electronic form), Hyderabad: University of Hyderabad.

3. Uma Maheshwar Rao, G., Chaithra, T.P., Santosh Jena. 2004. A Generic Architecture for Morphological Generators of Morphologically Complex Agglutinative Languages LECTURE COMPENDIUM, Symposium on Indian Morphology, Phonology & Language Engineering (SIMPLE'04) pp. 13-16. Kharagpur; Indian Institute of Technology.

4. Uma Maheshwar Rao, G. and Amba Kulkarni, P. Christopher, Mala. 2007. Morphological Analyzer and Its Functional Specifications for IL-ILMT System. CALTS, Hyderabad: University of Hyderabad.

5. Uma Maheshwar Rao, G. and Amba Kulkarni, P. 2006. Computer Applications in Indian Languages, Hyderabad: The centre for distance education, University of Hyderabad.

6. Uma Maheshwar Rao, G. and K. Parameswari.K. 2010. On the Description of Morphological Data for Morphological Analysers and Generators: A case of Telugu, Tamil and Kannada. In Mona Parekh (ed.) Morphological Analysers and Generators, pp73-81. Mysore: LDCIL,CIIL. www.ldcil.org/up/conferences/morph/presentation.html

7. Uma Maheshwar Rao, G. and M. Christopher, M. 2010. Word Synthesizer Engine. 2010. In Mona Parekh (ed.) Morphological Analysers and Generators, pp73-81. Mysore: LDCIL,CIIL. www.ldcil.org/up/conferences/morph/presentation.html

8. Parameswari. K. 2010. An Improvised Morphological Analyser cum Generator for Tamil: A Case of Implementing the Open Source Platform Apertium. In Mona Parekh (ed.) Morphological Analysers and Generators, pp124-131. Mysore: LDCIL,CIIL. www.ldcil.org/up/conferences/morph/presentation.html