A Simple Surface Realization Engine for Telugu

Sasi Raja Sekhar Dokkara, Suresh Verma Penumathsa
Dept. of Computer Science
Adikavi Nannayya University, India
dsairajasekhar@gmail.com,vermaps@yahoo.com

Somayajulu G. Sripada
Dept. of Computing Science
University of Aberdeen, UK
yaji.sripada@abdn.ac.uk

Abstract

Telugu is a Dravidian language with nearly 85 million first language speakers. In this paper we report a realization engine for Telugu that automates the task of building grammatically well-formed Telugu sentences from an input specification consisting of lexicalized grammatical constituents and associated features. Our realization engine adapts the design approach of SimpleNLG family of surface realizers.

1 Introduction

Telugu is a Dravidian language with nearly 85 million first language speakers. It is a morphologically rich language (MRL) with a simple syntax where the sentence constituents can be ordered freely without impacting the primary meaning of the sentence. In this paper we describe a surface realization engine for Telugu. Surface realization is the final subtask of an NLG pipeline (Reiter and Dale, 2000) that is responsible for mechanically applying all the linguistic choices made by upstream subtasks (such as microplanning) to generate a grammatically valid surface form. Our Telugu realization engine is designed following the SimpleNLG (Gatt and Reiter, 2009) approach which recently has been used to build surface realizers for German (Bollmann, 2011), Filipino (Ethel Ong et al., 2011), French (Vaudry and Lapalme, 2013) and Brazilian Portuguese (de Oliveira and Sripada, 2014). Figure 1 shows an example input specification in XML corresponding to the Telugu sentence (1).

vAlYlYu aMxamEna wotalo neVmmaxigA naduswunnAru.
(They are walking slowly in a beautiful garden.) (1)

```
<?xml version="1.0"encoding="UTF-8"standalone="no">
<document>
<sentence type=" " predicate-type="verbal" respect="no">
<nounphrase role="subject">
<head pos="pronoun" gender="human" number="plural" person="third" case-marker=" " stem="basic">
vAdu</head>
</nounphrase>
<nounphrase role="complement">
<modifier pos="adjective" type="descriptive" suffix="aEna">
aMxamu</modifier>
<head pos="noun" gender="nonmasculine" number="singular" person="third" casemarker="lo" stem="basic">
wota</head>
</nounphrase>
<verbphrase type=" ">
<modifier pos="adverb" suffix="gA">
neVmmaxi</modifier>
<head pos="verb" tense-mode="presentparticiple">
naducu</head>
</verbphrase>
</sentence>
</document>
```

Figure 1. XML Input Specification

2 Related Work

Several realizers are available for English and other European languages (Gatt and Reiter, 2009; Vaudry and Lapalme, 2013; Bollmann, 2011; Elhadad and Robin, 1996). Some general purpose realizers (as opposed to realizers built as part of an MT system) have started appearing for Indian languages as well. Smriti Singh et al. (2007) report a Hindi realizer that includes functionality for choosing post-position markers based on semantic information in the input. This is in contrast to the realization engine reported in the current paper which assumes that choices of constit-

Proceedings of the 15th European Workshop on Natural Language Generation (ENLG), pages 1–8,
Brighton, September 2015.

uents, root words and grammatical features are all preselected before realization engine is called. There are no realization engines for Telugu to the best of our knowledge. However, a rich body of work exists for Telugu language processing in the context of machine translation (MT). In this context, earlier work reported Telugu morphological processors that perform both analysis and generation (Badri et al., 2009; Rao and Mala, 2011; Ganapathiraju and Levin, 2006).

2.1 The SimpleNLG Framework

A realization engine is an automaton that generates well-formed sentences according to a grammar. Therefore, while building a realizer the grammatical knowledge (syntactic and morphological) of the target language is an important resource. Realizers are classified based on the source of grammatical knowledge. There are realizers such as FUF/SURGE that employ grammatical knowledge grounded in a linguistic theory (Elhadad and Robin, 1996). There have also been realizers that use statistical language models such as Nitrogen (Knight and Hatzivassiloglou, 1995) and Oxygen (Habash, 2000). While linguistic theory based grammars are attractive, authoring these grammars can be a significant endeavor (Mann and Matthiessen, 1985). Besides, non-linguists (most application developers) may find working with such theory heavy realizers difficult because of the initial steep learning curve. Similarly building wide coverage statistical models of language too is labor intensive requiring collection and analysis of large quantities of corpora. It is this initial cost of building grammatical resources (formal or statistical) that becomes a significant barrier in building realization engines for new languages. Therefore, it is necessary to adopt grammar engineering strategies that have low initial costs. The surface realizers belonging to the SimpleNLG family incorporate grammatical knowledge corresponding to only the most frequently used phrases and clauses and therefore involve low cost grammar engineering. The main features of a realization engine following the SimpleNLG framework are:

1. A wide coverage morphology module independent of the syntax module
2. A light syntax module that offers functionality to build frequently used phrases and clauses without any commitment to a linguistic theory. The large uptake of the SimpleNLG realizer both in the academia and in the industry shows that the light weight approach to syntax is not a limitation.
3. Using 'canned' text elements to be directly dropped into the generation process achieving wider syntax coverage without actually extending the syntactic knowledge in the realizer.
4. A rich set of lexical and grammatical features that guide the morphological and syntactic operations locally in the morphology and syntax modules respectively. In addition, features enforce agreement amongst sentence constituents more globally at the sentence level.

3 Telugu Realization Engine

The current work follows the SimpleNLG framework. However, because of the known differences between Telugu and English SimpleNLG codebase could not be reused for building Telugu realizer. Instead our Telugu realizer was built from scratch adapting several features of the SimpleNLG framework for the context of Telugu.

There are significant variations in spoken and written usage of Telugu. There are also significant dialectical variations, most prominent ones correspond to the four regions of the state of Andhra Pradesh, India – Northern, Southern, Eastern and Central (Brown, 1991). In addition, Telugu absorbed vocabulary (Telugised) from other Indian languages such as Urdu and Hindi. As a result, a design choice for Telugu realization engine is to decide the specific variety of Telugu whose grammar and vocabulary needs to be represented in the system. In our work, we use the grammar of modern Telugu developed by (Krishnamurti and Gwynn, 1985). We have decided to include only a small lexicon in our realization engine. Currently, it contains the words required for the evaluation described in section 4. This is because host NLG systems that use our engine could use their own application specific lexicons. More over modern Telugu has been absorbing large amounts of English vocabulary particularly in the fields of science and technology whose morphology is unknown. Thus specialized lexicons could be required to model the morphological behavior of such vocabulary. In the rest of this section we present the design of our Telugu realizer.

As stated in section 2.1, a critical step in building a realization engine for a new language is to review its grammatical knowledge to understand

the linguistic means offered by the language to express meaning. We reviewed Telugu grammar as presented in our chosen grammar reference by Krishnamurti and Gwynn (1985). From a realizer design perspective the following observations proved useful:

1. Primary meaning in Telugu sentences is mainly expressed using inflected forms of content words and case markers or postpositions than by position of words/phrases in the sentence. This means morpho-phonology plays bigger role in sentence creation than syntax.

2. Because sentence constituents in Telugu can be ordered freely without impacting the primary meaning of a sentence, sophisticated grammar knowledge is not required to order sentence level constituents. It is possible, for instance, to order constituents of a declarative sentence using a standard predefined sequence (e.g. Subject + Object + Verb).

3. Telugu, like many other Indian languages, is not governed by a phrase structure grammar, instead fits better into a Paninian Grammar Formalism (Bharati et al., 1995) which uses dependency grammar. This means, dependency trees represent the structure of phrases and sentences. At the sentence level verb phrase is the head and all the other constituents have a dependency link to the head. At the phrase level too, head-modifier dependency structures are a better fit.

4. Agreement amongst sentence constituents can get quite complicated in Telugu. Several grammatical and semantic features are used to define agreement rules. Well-formed Telugu sentences are the result of applying agreement rules at the sentence level on sentence constituents constructed at the lower level processes.

Based on the above observations we found that the SimpleNLG framework with its features listed in section 2.1 is a good fit for guiding the design of our Telugu realization engine. Thus our realization engine is designed with a wide coverage morphology module and a light-weight syntax module where features play a major role in performing sentence construction operations.

Having decided the SimpleNLG framework for representing and operationalizing the grammatical knowledge, the following design decisions were made while building our Telugu realizer (we believe that these decisions might drive design of realizers for any other Indian Language as well):

1. Use wx-notation for representing Indian language orthography (see section 3.1 for more details)
2. Define the tag names and the feature names used in the input XML file (example shown in Figure 1) adapted from SimpleNLG and (Krishnamurti and Gwynn, 1985) for specifying input to the realization engine. It is hoped that using English terminology for specifying input to our Telugu realizer simplifies creating input by application developers who usually know English well and possess at least a basic knowledge of English grammar. (see section 3.2 for more details)
3. In order to offer flexibility to application developers our realization engine orders sentence level constituents (except verb which is always placed at the end) using the same order in which they are specified in the input XML file. This allows application developers to control ordering based on discourse level requirements such as focus.
4. The grammar terminology used in our engine does not directly correspond to the Karaka relations (Bharati et al., 1995) from the Paninian framework because we use the grammar terminology specified by Krishnamurti and Gwynn (1985) which is lot closer to the terminology used in SimpleNLG. We are currently investigating opportunities to align our design lot closer to the Paninian framework. We expect such approach to help us while extending our framework to generate other Indian languages as well.

3.1 WX-Notation

WX notation (See appendix B in Bharati et al, 1995) is a very popular transliteration scheme for representing Indian languages in the ASCII character set. This scheme is widely used in Natural Language Processing in India. In WX notation the small case letters are used for un-aspirated consonants and short vowels while the capital case letters are used for aspirated consonants and long vowels. The retroflexed voiced and voiceless consonants are mapped to ‘t, T, d and D’. The dentals are mapped to ‘w, W, x and X’. Hence the name of the scheme “WX”, referring to the idiosyncratic mapping.

3.2 The Input Specification Scheme

The input to the current work is a tree structure specified in XML, an example is shown in Figure 1. The root node is the sentence and the nodes at the next level are the constituent phrases that have a role feature representing the grammatical functions such as subject, verb and complement performed by the phrase. Each of the lower level nodes could in turn have their own head and modifier children. Each node also can take attributes which represent grammatical or lexical features such as number and tense. For example the subject node in Figure 1 can be understood as follows:

```
<nounphrase role="subject">
<head
pos="pronoun"gender="human"number="p
lural"person="third"casemarker="   "
stem="basic">
vAdu</head>
</nounphrase>
```

This node represents the noun phrase that plays the role of subject in the sentence. There is only one feature, the head to the subject node whose type is nominative. The lexical features of the head "vAdu" are part-of-speech (*pos*) which is pronoun, person which is third person, number which is plural, gender which is human, and case marker which is null.

3.3 System Architecture

The sentence construction for Telugu involves the following three steps:

1. Construct word forms by applying morpho-phonological rules selected based on features associated with a word (word level morphology)

2. Combine word forms to construct phrases using 'sandhi' (a morpho-phonological fusion operation) if required (phrase building)

3. Apply sentence level agreement by applying agreement rules selected based on relevant features. Order sentence constituents following a standard predefined sequence. (sentence building)

Our system architecture is shown in Figure 2 which involves morphology engine, phrase builder and sentence builder corresponding to these three steps. The rest of the section presents how the example sentence (1) is generated from the input specification in Figure 1.

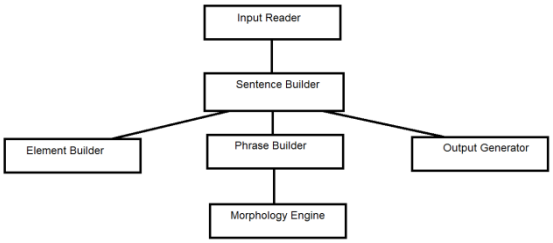

Figure 2. System Architecture

3.4 Input Reader

The Input Reader is the module which acts as an interface between the sentence builder and the input. Currently the input reader accepts only our XML input specification but in the future we would like to extend it to accept other input specifications such as SSF (Bharati et al., 2007). This module ensures that the rest of the engine receives input in the required form.

3.5 Sentence Builder

The Sentence Builder is the main module of the current system which has a centralized control over all the other modules. It performs four subtasks:

1. Sentence Builder first checks for predefined grammatical functions such as subject, object, complement, and verb which are defined as features of the respective phrases in the input. It then calls the appropriate element builder for each of these to create element objects which store all the information extracted from the XML node.

2. These element objects are then passed to appropriate phrase builder to receive back a string which is the phrase that is being constructed according to the requirements of the input.

3. After receiving all the phrases from the appropriate phrase builders the Sentence Builder applies the agreement rules. Since Telugu is nominative-accusative language the verb agrees with the argument in the nominative case. Therefore the predicate inflects based on the gender, person and number of the noun in the nominative case. There are three features at the sentence level namely *type*, *predicate-type*, and *respect*. The feature *type* refers to the type of the sentence. The current work handles only simple sentences therefore

it is not set to any value. The feature *predicate-type* can have any one of the three values namely *verbal*, *nominative*, and *abstract*. The feature *respect* can have values *yes* or *no*. The agreement also depends on the features *predicate-type*, and *respect*.

4. Finally, the sentence builder orders the phrases in the same order they are specified in the input.

In the case of the example in Figure 1 the sentence builder finds three grammatical functions - one finite verb, one locative complement, and one nominative subject. In the example input (1) the values for the feature *predicate-type* is "*verbal*" and for *respect* is "*no*". The Sentence Builder retrieves appropriate rule from an externally stored agreement rule base. In the example input (1) where *predicate-type* is set to *verbal*, the *number* of the subject is *plural* and the *gender* is *human* the Sentence Builder retrieves the appropriate suffix "nnAru". This suffix is then agglutinated to the verb "naduswu" which is returned by the morphology engine to generate the final verb form, "naduswunnAru" with the required agreement with subject.

"naduswu"+ "nnAru"----→ "naduswunnAru"

After the construction of the sentence the Sentence Builder passes it to the Output Generator which prints the output.

3.6 Element Builder

The element builder of each grammatical function checks for lower level functions like head and modifier and calls the appropriate element builder for the head and modifier which converts the lexicalized input into element objects with the grammatical constituents as their instance variables and returns the element objects back to the Sentence Builder. Our realizer creates four types of element objects namely SOCElement, VAElement, AdjectiveElement, and AdverbElement. The SOCElement represents the grammatical functions subject, object and complement. The subject in the example of (1) is "vAdu" for which a SOCElement is created with the specified features. Similarly a SOCElement is created for the complement "wota" and its modifier "aMxamu" which is an AdjectiveElement. Finally a VAElement is created for the verb "naducu" and the modifier "neVmmaxi" which is an AdverbElement.

3.7 Phrase Builder

Telugu sentences express most of the primary meaning in terms of morphologically well-formed phrases or word groups. In Telugu the main and auxiliary verbs occur together as a single word. Therefore their generation is done by the morphology engine. Telugu sentences are mainly made up of four types of phrases - Noun Phrase, Verb Phrase, Adjective Phrase, and Adverb Phrase. Noun phrases and verb phrases are the main constituents in a sentence while the Adjective Phrase and the Adverb Phrase only play the role of a modifier in a noun or verb phrase. There is one feature at the Noun Phrase level "role" which specifies the role of the Noun Phrase in the sentence. The phrase builder passes the elements constructed by the element builder to the morphology engine and gets back the respective phrases with appropriately inflected words. In the example input in (1), there are three constituent phrases, viz, two noun phrases for subject and complement and a verb phrase. One of the noun phrases also contains an adjective phrase which is an optional modifying element of noun heads in head-modifier noun phrases. The adjective phrase may be a single element or sometimes composed of more than one element. The verb phrase also contains an adverb phrase which is generally considered as a modifier of the verb. The phrase builder passes five objects i.e., two SOCElement objects, one AdjectiveElement object, one VAElement object, and one AdverbElement object to the morphology engine and gets back five inflected words which finally become three phrases, viz, two noun phrases "vAlYlYu", "aMxamEna wotalo", and one verb phrase "neVmmaxigA naduswu".

3.8 Morphology Engine

The morphology engine is the most important module in the Telugu realization engine. It is responsible for the inflection and agglutination of the words and phrases. The morphology engine behaves differently for different words based on their part of speech (*pos*). The morphology engine takes the element object as the input, and returns to the phrase builder the inflected or agglutinated word forms based on the rules of the language. In the current work morphology engine is a rule based engine with the lexicon to account for exceptions to the rules. The rules used by the morphology engine are stored in external files to allow changes to be made externally.

3.8.1 Noun

Noun is the head of the noun phrase. Telugu nouns are divided into three classes namely (i) proper nouns and common nouns, (ii) pronouns, and (iii) special types of nouns (e.g. numerals) (Krishnamurti and Gwynn, 1985). All nouns except few special type nouns have *gender*, *number*, and *person*. Noun morphology involves mainly plural formation and case inflection. All the plural formation rules from sections 6.11 to 6.13 of our grammar reference have been implemented in our engine.

The head of the complement in the example (1) has one noun "wotalo". The word "wota" along with its feature values can be written as follows:

"wota", noun, nonmasculine, singular, third, basic, "lo"---wotalo

The formation of this word is very simple because the word "wota" in its singular form and the case marker "lo" get agglutinated through a sandhi (a morpho-phonological fusion operation) formation as follows:

'wota'+lo----- wotalo

3.8.2 Pronoun

Pronouns vary according to *gender*, *number*, and *person*. There are three persons in Telugu namely *first*, *second*, and *third*. The *gender* of the nouns and pronouns in Telugu depend on the *number*. The relation between the *number* and *gender* is shown in table 1.

Number	Gender
singular	masculine, nonmasculine
plural	human, nonhuman

Table1: Relationship between number and gender

Plural formation of pronouns is not rule based. Therefore they are stored externally in the lexicon. The first person pronoun "nenu" has two plural forms "memu" which is the exclusive plural form and "manamu" which is the inclusive plural form. In the generation of the plural of the first person a feature called "exclusive" has to be specified with the value "yes", or "no". Along with *gender*, *number*, and *person* there is one more feature which is *stem*. The stem can be either *basic* or *oblique*. The formation of the pronoun "vAlYlYu" in the example of (1) which is the head of the subject along with its feature values can be written as follows:

"vAdu", pronoun, human, plural, third, basic,""-vAlYlYu

In this case the stem is *basic*. The *gender* of the pronoun is *human* because the *number* is *plural* as mentioned in table 1. The word "vAlYlYu" is retrieved from the lexicon as the plural for the word "vAdu" and the feature values.

3.8.3 Adjective

Adjectives occur most often immediately before the noun they qualify. The basic adjectives or the adjectival roots which occur only as adjectives are indeclinable (e.g. oka (one), ara (half)). Adjectives can also be derived from other parts of speech like verbs, adverbs, or nouns. The adjective "aMxamEna" in the example of (1) is a derived adjective formed by adding the adjectival suffix "aEna" to the noun "aMxamu". The formation of the word "aMxamEna" in the example (1) along with its feature values can be written as follows:

"aMxamu", adjective, descriptive,"aEna"--aMxamEna

The current work does not take into consideration the type of an adjective and will be included in a future version. The formation of this word is again through a sandhi formation as follows:

aMxamu+aEna-------- aMxamEna

Here the sandhi formation eliminates the "u" in the first word; "a" in the second word and the word "aMxamEna" is formed.

3.8.4 Verb

Telugu verbs inflect to encode gender, number and person suffixes of the subject along with tense mode suffixes. As already mentioned gender, number and person agreement is applied at the sentence level. At the word level, verb is the most difficult word to handle in Telugu because of phonetic alterations applied to it before being agglutinated with the tense-aspect-mode suffix (TAM). Telugu verbs are classified into six classes (Krishnamurti, 1961). Our engine implements all these classes and the phonetic alternations ap-

plicable to each of these classes are stored externally in a file.

The verb in the example of Figure 1 has one verb "naducu" along with its feature values. The formation of the verb "naduswu" can be written as follows:

"naducu",verb, present participle------naduswu

The word "naducu" belongs to class IIa, for which the phonetic alteration is to substitute "cu" with "s", and therefore the word gets inflected as follows:

naducu----------------nadus

The tense mode suffix for present participle is "wu", and the word becomes "naduswu". The gender and number of the subject also play a role in the formation of the verb which is discussed in section 3.5.

3.8.5 Adverb

All adverbs fall into three semantic domains, those denoting time, place and manner (Krishnamurti and Gwynn 1985). The adverb "neVmmaxigA" in the example (1) is a manner adverb as it tells about the way they are walking "neVmmaxigA naduswunnaru (walking slowly)". In Telugu manner adverbs are generally formed by adding "gA" to adjectives and nouns. The formation of the adverb "neVmmaxigA" in the example (1) along with its feature values can be written as follows:

"neVmmaxi", adverb,"gA"----------neVmmaxigA

The formation of this word is a simple sandhi formation.

3.9 Output Generator

Output Generator is the module which actually generates text in Telugu font. The Output generator receives the constructed sentence in WX-notation and gives as output a sentence in Telugu based on the Unicode Characters for Telugu.

4 Evaluation

The current work addresses the problem of generating syntactically and morphologically well-formed sentences in Telugu from an input specification consisting of lexicalized grammatical constituents and associated features. In order to test the robustness of the realization engine as the input to the realizer changes we initially ran the engine in a batch mode to generate all possible sentence variations given an input similar to the one shown in Figure 1. In the batch mode the engine uses the same input root words in a single run of the engine, but uses different combinations of values for the grammatical features such as tense, aspect, mode, number and gender in each new run. Although the batch run was originally intended for software quality testing before conducting evaluation studies, these tests showed that certain grammatical feature combinations might make the realization engine produce unacceptable output. This is an expected outcome because our engine in the current state performs very limited consistency checks on the input.

The purpose of our evaluation is to measure our realizer's coverage of the Telugu language. One objective measure could be to measure the proportion of sentences from a specific text source (such as a Telugu newspaper) that our realizer could generate. As a first step towards such an objective evaluation, we first evaluate our realizer using example sentences from our grammar reference. Although not ideal this evaluation helps us to measure our progress and prepares us for the objective evaluation. The individual chapters and sections in the book by Krishnamurti and Gwynn (1985) follow a standard structure where every new concept of grammar is introduced with the help of a list of example sentences that illustrate the usage of that particular concept. We used these sentences for our evaluation. Please note that we collect sentences from all chapters. This means our realizer is required to generate for example verb forms used in example sentences from other chapters in addition to those from the chapter on verbs. A total of 738 sentences were collected from chapter 6 to chapter 26, the main chapters which cover Telugu grammar. Because the coverage of the current system is limited, we don't expect the system to generate all these 738 sentences. Among these, 419/738 (57%) sentences were found to be within the scope of our current realizer. Many of these sentences are simple and short. For each of the 419 selected sentences our realizer was run to generate the 419 output sentences. The output sentences matched the original sentences from the book completely. This means at this stage we can quantify the coverage of our realizer as 57%

(419/738) against our own grammar source. A more objective measure of coverage will be estimated in the future.

Having built the functionality for the main sentence construction tasks, we are now in a good position to widen the coverage. Majority of the remaining 319 sentences (=738-419) involve verb forms such as participles and compound verbs and medium to complex sentence types. As stated above, we intend to use this evaluation to drive our development. This means every time we extend the coverage of the realizer we will rerun the evaluation to quantify the extended coverage of our realizer. The idea is not to achieve 100% coverage. Our strategy has always been to select each new sentence or phrase type to be included in the realizer based on its utility to express meanings in some of the popular NLG application domains such as medicine, weather, sports and finance.

5 Conclusion

In this paper, we described a surface realizer for Telugu which was designed by adapting the SimpleNLG framework for free word order languages. We intend to extend the current work further as stated below:

1. Extend the coverage of our realizer and perform another evaluation to characterize the coverage of the realizer more objectively.
2. Create a generalized framework for free word order language generation (specifically for Indian languages). The existing framework could be used to generate simple sentences from other Indian languages by plugging in the required morphology engine for the new language.

Reference

Albert Gatt and Ehud Reiter "*SimpleNLG: A realization engine for practical applications*", Proceedings of ENLG 2009, pages 90-93, 2009.

AksharaBharati, Vineet Chaitanya, Rajeev Sangal "*Natural Language Processing A Paninian Perspective*" Prentice-Hall of India, New Delhi, 1995.

Akshara Bharati, Rajeev Sangal, Dipti M Sharma "*SSF: Shakti Standard Format Guide*" LTRC, IIIT, Hyderabad, Report No: TR-LTRC-33, 2007.

Benoit Lavoie and Owen Rambow "*A Fast and Portable Realizer for Text Generation Systems*" Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP97), Washington, 1997.

BH. Krishnamurti and J P L Gwynn, "*A Grammar of Modern Telugu*" Oxford University Press, 1985.

BH. Krishnamurti "*Telugu Verbal Bases a comparative and Descriptive Study*" University of California Press Berkley & Los Angeles, 1961.

Brown, C.P., "*The Grammar of the Telugu Language*". New Delhi: Laurier Books Ltd, 1991.

Elhadad M. & Robin J. (1996). "*A reusable comprehensive syntactic realization component"*. Paper presented at Demonstrations and Posters of the 1996 International Workshop on Natural Language Generation (INLG '96), Herstmonceux, England.

Ethel Ong, Stephanie Abella, Lawrence Santos, and Dennis Tiu "*A Simple Surface Realizer for Filipino* " 25th Pacific Asia Conference on Language, Information and Computation, pages 51–59, 2011.

HabashN. (2000). *OxyGen: "A Language Independent Linearization"*Engine. Paper presented at AMTA. London: Ablex. Available as USC/ISI Research Report RR-83-105.

Knight K. and V. Hatzivassiloglou. NITROGEN: "*Two-Level, Many-Paths Generation*". Proceedings of the ACL-95 conference. Cambridge, MA 1995.

MadhaviGanapathiraju and Lori Levin "*TelMore: Morphological Generator for Telugu Nouns and Verbs*", 2006.

Mann, W.C. and C.M.I.M. Matthiessen. Nigel: "*A Systemic Grammar for Text Generation*". In R. Benson and J. Greaves (eds), Systemic perspectives on Discourse: Proceedings of 9th International Systemics workshop 1985.

Marcel Bollmann, "*Adapting SimpleNLG to German*" Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), pages 133–138, Nancy, France, September 2011.

Pierre-Luc Vaudry and Guy Lapalme "*Adapting SimpleNLG for bilingual English-French realisation*" Proceedings of the 14th European Workshop on Natural Language Generation, pages 183–187, Sofia, Bulgaria, August 8-9 2013.

Rodrigo de Oliveira, Somayajulu Sripada "*Adapting SimpleNLG for Brazilian Portuguese realisation*", 2014.

Smriti Singh, MrugankDalal, Vishal Vachhani, Pushpak Bhattacharyya, Om P. Damani "*Hindi Generation from Interlingua (UNL)*" in Proceedings of MT summit, 2007.

Sri BadriNarayanan.R, Saravanan.S, Soman K.P "*Data Driven Suffix List and Concatenation Algorithm for Telugu Morphological Generator*" International Journal of Engineering Science and Technology (IJEST), 2009.

Uma MaheshwarRao, G. and Christopher Mala "*TELUGU WORD SYNTHESIZER*" International Telugu Internet Conference Proceedings, Milpitas, California, USA 28th -30th September, 2011.