International Conference on Machine Learning and Computational Intelligence-2017
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 7 | ISSN : 2456-3307

Empirical Analysis of Context Sensitive Grammars and Parse Trees for Disambiguiting Telugu Language Sentences

Jinka Sreedhar*[1], SK Althaf Hussain Basha[1], D. Praveen Kumar[2], A. Jagan[3] , Baijnath Kaushik[4]
[1]Department of Computer Science and Engineering Gokaraju Rangaraju Institute of Engineering and Technoloy, Hyderabad, India
[2] Department of Computer Science and Engineering Institute of Technology, Dhanbad, Jharkhand, India
[3]Department of Computer Science and Engineering B.V.Raju Institute of Technology, Narsapur, Telangana, India
[4]Department of Computer Science and Engineering Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir, India

ABSTRACT

This research paper explores the impact of Context Sensitive Grammars (CSG) and Parse Trees for construction of a Telugu Language Sentences. Based on the CSG Rules here we derived the derivations for the respective strings. Later we constructed the Parser Trees for the above said strings. Finally we analysed whether the string is ambiguous or unambiguous. Here for analysis we considered the Large Scale Open Source Telugu carpus. The main aim of finding out these methods, is to find out solution to the problem of ambiguity in Telugu Language Sentences. Here designing of Context Sensitive Grammars Rules and Parse Trees are explained with examples.

Keywords : Natural Language Processing (NLP), Context Sensitive Grammars (CSG) Rules, Parse Trees(PT's), Derivations.

I. INTRODUCTION

The syntax of a language may be specified using a notation called Context Sensitive Grammar (CSG). A context sensitive grammar consists of terminals, non-terminals, a start symbol and production rules. The set of tokens are called the terminal symbols. These are the basic symbols from which strings are formed. Non terminals are the symbols which represent syntactic variables that denote sets of strings. They do not exist in the source program they only help in defining the language generated by the grammar. One of the non-terminals designated as the start symbol. We shall follow the convention of listing the production for the start symbol. The set of strings denoted by the start symbol is the language defined by the grammar. A production rule has a non-terminal symbol on the left hand side followed by an arrow and a sequence of symbols on the right side. This sequence of symbols may contain a combination of terminals and non-terminals[9,11,13].

The organization of this paper is as follows: Section II describes the CSG and its notations, Section III case study IV deals with derivations of CSG Grammar, Section V explores the Parser Trees , Section VI shows the acknowledgements and Section VII deals with conclusion and future enhancement followed by the references.

II. CONTEXT SENSITIVE GRAMMARS

We may have more than one production rule for the same non terminal. In that case, we can group their right hand side by using symbol | to separate the alternate right hand side. The Context Sensitive Grammar explains the sensitive nature of words by its applications. In general a CSG [10,12,17] is a set of recursive rewriting rules called productions that are used to generate patterns of strings and it consists of the following components:

- ✓ A finite set of terminal symbols (Σ).
- ✓ A finite set of non-terminal symbols (NT).
- ✓ A finite set of productions (P).
- ✓ A start symbol (S).

Let G be a Context Sensitive Grammar for which the production rules are:

$$S \Rightarrow NP\ \ VP$$
$$N\ NP \Rightarrow N\ \ NP\ |QC\ \ NP|\,SYM$$
$$V\ VP \Rightarrow VP\ \ NP\ |V\ \ SYM|\,SYM$$
$$NP\ \ VP\ \ \Rightarrow VP$$

Figure 2.1. Context Sensitive Grammar

Where,S is a Sentence, NP is a Noun Phrase, VP is a Verb Phrase, N is a Noun, V is a Verb, QC is a Cardinal, SYM is a Sym.

III. CASE STUDY

In this method of explaining the possibility of avoiding the ambiguity in the structure of Telugu language sentence, one Telugu Sentence has been taken to explain how this arithmetical method has worked out. For example, the sentence is :

1. rEwulu edAxiki mUdu kArla paMtalu paMdiswAru .
   N N QC N N V SYM

Here, in the Telugu sentence 1, each part is segregated and named as N,N,QC,N,N,N,V and SOV and the POS tags explain how this method eliminates the ambiguity of the sense of the sentence when it is applied.

All these views have been taken from the Morphological Analyzer as an example and certain rules have been set in this process of explaining how a sense ambiguity can be avoided in the Telugu language sentence structure.

From sentence 1 the word kAru is taken to explain the meaning of one which is used only in the region of Rayalaseema which has its own dialect and a similar meaning cannot be seen in the two other regions of Telugu speaking states, namely, Andhra and Telangana.

kAru comes under the category of noun. In taking this word as a noun, its meaning has been lost. It is a moving vehicle. In this particular context though it plays the role if an adjective which tells the number, in this context it is considered a Noun.

IV. DERIVATIONS

Here Derivation provides a means for generating the sentences of a language. If one chooses the leftmost non-terminal in a given sentential form then it is called leftmost derivation. If one chooses the rightmost non-terminal in a given sentential form then it is called rightmost derivation. Derivation from S means generation of string w from S. Any language construct can be defined by the CSG [3,15,16]. The above grammar generates different strings by providing many sentential forms as shown below.

$$
\begin{aligned}
& S \Rightarrow NP\ \ VP \\
& \Rightarrow NN\ \ NP\ VP \\
& \Rightarrow N\ \ NP\ VP \\
& \Rightarrow\ N\ \ NN\ \ NP\ VP \\
& \Rightarrow N\ \ NNP\ VP \\
& \Rightarrow N\ \ N\ \ QC\ \ NP\ VP \\
& \Rightarrow NN\ \ QC\ \ NN\ \ NP\ VP \\
& \Rightarrow N\ \ N\ \ QC\ \ N\ \ N\ \ NP\ VP \\
& \Rightarrow N\ \ N\ \ QC\ \ N\ \ \ N\ \ NP\ VP \\
& \Rightarrow N\ \ N\ \ QC\ \ N\ \ N\ \ VP \\
& \Rightarrow N\ \ N\ \ QC\ \ N\ \ N\ \ V\ \ SYM
\end{aligned}
$$

Figure 4.1. Derivation for the input sentence 1

As an explanation of these derivations, the first one is a sentence deriving Noun Phrase and Verb Phrase, and by taking the Noun Phrase the Noun and Noun Phrase have been derived.

As a second step of derivation, the Noun Phrase has been taken and from it Noun and Noun Phrase have been derived. At the third stage Noun phrase and Verb phrase have been derived. From Noun Phrase, Noun and Noun Phrase have been derived.

Now from the Verb Phrase only the verb has been derived.

From the Figure 4.1, the following are clarified so as to root out the ambiguity in the sentence word order. They are:

N N QC N N V SYM

A different representation of the Parse Tree is given to explain how this ambiguity in the sentence word order can be avoided.

V. PARSE TREES

A parse tree [1,4,5] is an equivalent form of showing a derivation which represents a derivation graphically or pictorially. A parse-tree is an internal structure, created by the compiler or interpreter while parsing some language construction. Parsing is also known as 'syntax analysis'.

A parse tree for a grammar G is a tree where

- ✓ the root is the start symbol for G
- ✓ the interior nodes are the non-terminals of G
- ✓ the leaf nodes are the terminal symbols of G.
- ✓ the children of a node T (from left to right) correspond to the symbols on the right hand side of some production for T in G.

Every terminal string generated by a grammar has a corresponding parse tree; every valid parse tree represents a string generated by the grammar (called the yield of the parse tree).

In this parse tree method of explaining, the first one is S which stands as the root of the parse tree. From this S, the NP and VP have been used to construct the word order. From NP, NN and NP have been taken to derive the Noun, ie. N. Again NP is taken to explain NN and NP, and derive the Noun, ie. N. Again, in the similar manner, NP has been taken directly to derive the Noun, ie. N. Now to derive a verb, Verb Phrase has been taken, all these explained in the Figure 5.1.

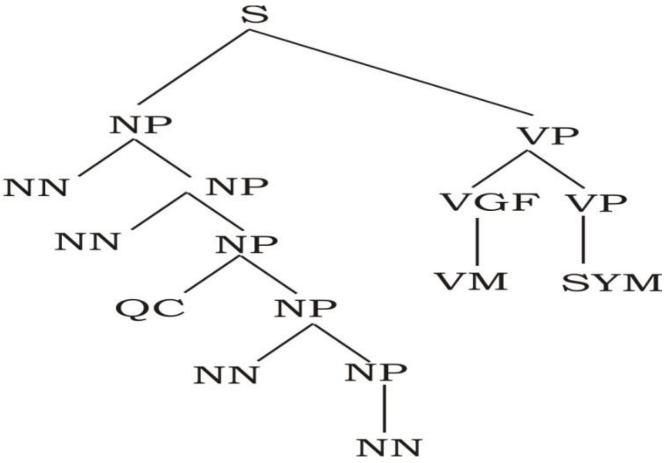

Figure 5.1. Parse tree for Input sentence 1

All these methods explain clearly how a sentence structure can be changed to get the proper sense and to avoid the ambiguity in all kinds of word order.

VI. ACKNOWLEDGMENTS

We are very thankful to all the esteemed authors in a reference list, to make this research article in a better shape and in right direction.

VII. CONCLUSION & FUTURE ENHANCEMENT

Here we described about the impact of noun ambiguity has been analyzed in Telugu Language Sentences empirically. The noun ambiguity in the Telugu Language Sentences is rooted out by applying the Context Sensitive Grammar Rules. There is a scope for further research on verbs, adjectives and adverbs to measure their impact.

VIII. REFERENCES

[1]. Aho, A.V., and Johnson, S.C.1974]. "LR parsing," Computing Surveys 6:2, 99-124.

[2]. Aho, A.V., and Johnson, S.C. , and Ullman, J. D.1975]."Deterministic parsing of ambiguous grammars," Comm. ACM 18:8, 441-452.

[3]. Aho, A.V., and Peterson, T.G,1972]. "A minimum distance error correcting parser for context-free languages," SIAM J. Computing 1:4, 305-312.

[4]. Aho, A.V., and Ullman, J.D. 1972b]. The Theory of Parsing Translation and Compiling, Vol. I:Parsing, Prentice-Hall, Englewood Cliffs, N. J.

[5]. Aho, A.V., and Ullman, J.D. 1972c]. "Optimization of LR(k) parsers," J. Computer and systems Sciences 6:6, 573-602.

[6]. Aho, A.V., and Ullman, J.D.1972a]. The Theory of Parsing, Translation and Compiling, Vol. II: Compiling, Prentice- Hall, Englewood Cliffs, N. J.

[7]. Aho, A.V., and Ullman, J.D.1972b]. " A technique for speeding up LR(k) parsers." SIAM J. Computing 2:2, 106-127.

[8]. Anderson, J. P. 1964]. "A note on some compiling algorithms," comn. ACM 7:3, 149-153.

[9]. Anderson, T., Eve, J., and Horning, J. J.1973]. " Efficient LR(1) paresers," Acta Informatica 2:1, 12-39.

[10]. Backhouse, R.C. 1976]. "An alternative approach to the improvement of LR parsers," Acta Informatica 6:3, 277-296.

[11]. Bar Hillel, Y., Perles, M., and Shamir, E. 1961]. "On formal properties of simple phrase structure grammers," Z. Phonetik, Sprachwissenschaft und Kommunikationsforschung 14, pp. 143-172.

[12]. Barnard, D. T. 1975]. "A survey of syntax error handling techniques," Computer Science Reaserch Group, Univ. of Toronto, Toronto, Ont., Canada.

[13]. Birman, A., and Ullman, J.D. 1973]. "Parsing algorithms with backtrack," Information and Control 23:1, 1-34.

[14]. Bochmann, G. V. 1976]. "Semantic evaluation from left to right," Comm. ACM 19:2, 55-62.

[15]. Brzozowiski, J. A. 1964]. "Derivatives of regular expressions," J. ACM 11:4, 481-488.

[16]. Cheatham, T. E. Jr., and Sattley, K. 1964]. "Syntax directed compiling," Proc. AFIPS 1964 Spring Joint Computer Conf. Spartan Books, Baltimore Md., 31-57.

[17]. Chomsky, N. 1959]. "On Certain formal properties of grammers," Information and Control 2:2, 137-167.