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

NLP : Context Free Grammars and Parse Trees for Disambiguiting Telugu Language Sentences

Jinka Sreedhar*[1], Dr Sk Althaf Hussain Basha[2] , D. Praveen Kumar[3], A. Jagan[4], Baijnath Kaushik[5]

[1]Department of Computer Science and Engineering, Gokaraju Rangaraju Institute of Engineering and Technoloy, Hyderabad, India

[2]Professor and Vice Principal, Department of Computer Science and Engineering, A1GIET, Markapur, Andhra Pradesh, India

[3]Department of Computer Science and Engineering Indian Institute of Technology, Dhanbad, Jharkhand, India

[4]Department of Computer Science and Engineering, B.V.Raju Institute of Technology, Narsapur, Telangana, India

[5]Department of Computer Science and Engineering, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir, India

ABSTRACT

Several studies have explained the benefits of using Context Free Grammars(CFGs) of derivations and Parse Trees to reduce the ambiguity in Natural Language Sentences. However, these benefits are dependent on the CFG Rules and Derivation steps. This research paper explains the power of CFGs and Parse Trees for construction of a Telugu Language Sentences. Based on the CFG 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 we considered the Large Scale Open Source Telugu carpus for analysis.

Keywords : CFG, Parse Trees, Derivations, Telugu Corpus

I. INTRODUCTION

The syntax of a language may be specified using a notation called Context Free Grammar (CFG). A Context Free 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 CFG and its notations, Section III deals with derivations of CFG Grammar, Section IV explores the Parser Trees , Section V shows the acknowledgements and Section VI deals with conclusion followed by the references.

II. CONTEXT FREE GRAMMARS

Here Context Free Grammar rules and regulations in Telugu language are explained. From this Context Free Grammar Derivations and Parse Trees are for the given sentence are derived[6,7,8]. These methods resolve the problem of ambiguity and help in the understanding of the sense of the sentence without any misunderstanding.

The set of tokens are called the terminals from which strings are composed. Non-terminals represent syntactic variables that denote sets of strings. They only help in defining the language generated by the grammar [9]. The strings denoted by start symbol constitute language as defined by grammar.

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. CFG, sometimes called a phrase structure grammar[2] plays a central role in the description of natural languages. In general a CFG [10,11,12,17] 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 Free Grammar for which the production rules are:

3.1 Methodology for Derivations

1. waMdri ceVppina viRayAlu AlociMcevAdu.
   n v n pn

Here, in the sentence 1, as an example there is a noun phrase and a verb phrase and noun phrase (NP) has been taken to find out noun (n). The ambiguity can be cleared by explaining the sentence 1 in the Figure 3.1.

$$S \Rightarrow NP\ VP$$
$$NP \Rightarrow Noun \mid VP\ PP \mid \text{Pr}onoun$$
$$VP \Rightarrow Verb \mid NP\ PP \mid VP\ PP \mid \varepsilon$$
$$PP \Rightarrow NP\ PP \mid NP\ PP \mid VP\ PP \mid \varepsilon$$
$$Noun \Rightarrow n$$
$$Verb \Rightarrow v$$
$$\text{Pr}onoun \Rightarrow pn$$

Figure 2.1. Context Free Grammar

Where, S stands for Sentence, NP stands for Noun Phrase, VP stands for Verb Phrase, PP stands for Prepositional Phrase, Pn stands for pronoun, n stands for noun, v stands for verb.

III. 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 CFG [3,15,16]. The above grammar generates different strings by providing many sentential forms as shown below.

In the sentence 1, for example, will explain how these POS change for different purposes. Just in order to explain this method, this derivation is explored.

S => NP VP
⇨ Noun VP
⇨ n VP PP
⇨ n Verb PP
⇨ n v NP PP
⇨ n v Noun PP
⇨ n v n NP PP
⇨ n v n Pn PP
⇨ n v n pn €

n v n Pn €

n v n pn

Figure 3.1. Derivation of "n v n pn"

The start symbol of the above grammar is S. Any grammar contains terminals and non-terminals. The non-terminal symbol occurs at the left hand side. These are the symbols which need to be expanded. The non-terminals are replaced by the terminals which it derives.

The above string is derived from S step by step as follows:

- ✓ First the nonterminal NP present at the left side is replaced by its substring noun.
- ✓ Then it is substituted by its substring n.
- ✓ Then VP is substituted by its substring VP PP.
- ✓ Then again VP is substituted by its substring Verb.
- ✓ Then that Verb is substituted by its substring v.
- ✓ Then PP is substituted by its substring NP PP.
- ✓ Then again NP is substituted by its substring Noun, and then Noun is substituted by its substring n.
- ✓ Then again PP is substituted by its substring NP PP.
- ✓ Then again NP is substituted by its substring Pronoun.
- ✓ Finally, PP is substitued by its substring €.
- ✓ So that, finally we obtain the string.

S-> n v n Pn € from S-> NP VP

S => NP VP
⇨ Noun VP
⇨ n VP PP
⇨ n € PP
⇨ n NP PP
⇨ n pn PP
⇨ n pn NP PP
⇨ n pn Noun PP
⇨ n pn n PP
⇨ n pn n VP PP
⇨ n pn n Verb PP
⇨ n pn n v PP
⇨ n pn n v €
⇨ n pn n v

n pn n v

Figure 3.2. Derivation of "n pn n v"

The above string is derived from S step by step as follows:

- ✓ The non-terminal NP present at the left side is replaced by its substring noun.
- ✓ Then it is substituted by its substring n.
- ✓ Then VP is substituted by its substring VP PP.
- ✓ Then again VP is substituted by its substring €.
- ✓ € means null value, so we can just eliminate it.
- ✓ Then PP is substituted by its substring NP PP.
- ✓ Then NP is substituted by its substring pronoun (pn).
- ✓ Then again PP is substituted by its substring NP PP.
- ✓ Then again NP is substituted by its substring Noun, and then Noun is substituted by its substring n.
- ✓ Then again PP is substituted by its substring VP PP.
- ✓ Then again VP is substituted by its substring Verb ,and then Verb is substituted by one of the substring v.
- ✓ Finally, PP is substituted by its substring €.
- ✓ € means null value, so we can just eliminate it.
- ✓ So that, finally we obtain the string

S-> n pn n v from S-> NP PP

IV. 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'[13,14].

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).

Parse Trees for Sentence 1:

Consider the below grammar, implementing the parse tree for the strings generated by this grammar.

S => NP VP

NP => Noun | Pronoun

VP => Verb | VP PP| ϵ

PP => NP PP|VP PP| ϵ

Noun => n

Verb => v

Pronoun =>pn

Figure 4.1. Context Free Grammar

1) This grammar generates the string n v n pn. The parse tree for this string using CFG is as following steps.
2) Create a root labeled with S.
3) For each sentential form αi in the derivation, i ≥ 2, construct a parse tree whose yield is αi ,

We can use induction for constructing the for αi , given the tree for αi-1 as given below:

a. The tree for α1 = S is a single node labeled S.
b. Let αi-1 = X1 X2 …. Xr and αi is derived from αi-1 by replacing Xj by β = Y1 Y2 ….. Yk.

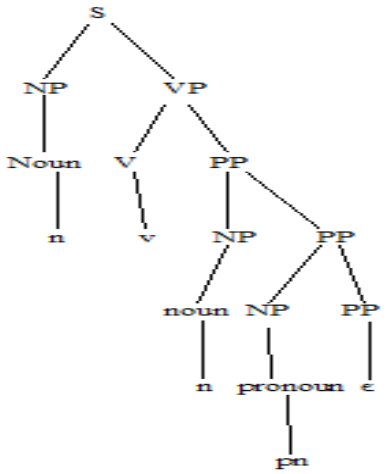

Figure 4.2. Parse Tree for "nvnpn"

S is a start symbol which derives NP VP, NP is a non-terminal which is substituted by noun and it is in turn substituted by the terminal n.

Now VP derives VP PP, PP with NP PP. NP is substituted by noun and with n.

Similarly PP derives NP PP and NP with the terminal pn. Finally, we obtain the string n v n pn.

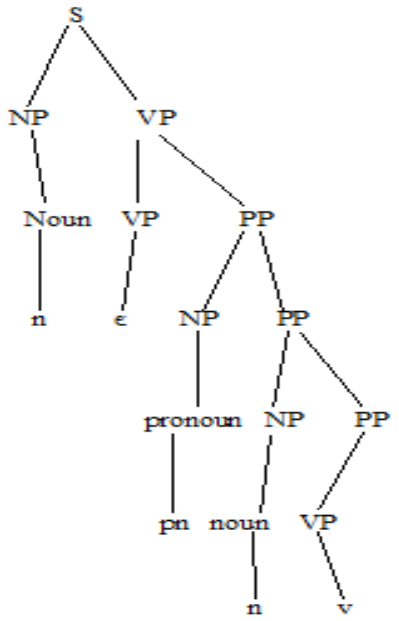

Figure 4.3. Parse Tree for "npnnv"

S is the start symbol for the above grammar which derives NP PP. NP is reduced to noun and inturn by n.

VP derives VP PP and PP to NP PP. Now NP is reduced to pronoun and to the terminal pn. Next PP is substituted by NP PP where NP to noun and PP to VP. Finally, we obtain the string n pn n v.

V. 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.

6. CONCLUSION and FUTURE ENHANCEMENT

Here we described about the Context Free Grammars, Derivations and Parse Trees. We observed the ambiguity between the Telugu Language Sentences. There is a scope for further research on other Natural Languages of nouns, verbs, adjectives and adverbs to measure their impact

VI. 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.