ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Efficient Hierarchical Parsing Technique Using LR

Poonam Nandal, Deepa Bura and Meeta Singh
Department of Computer Science & Engineering
Faculty of Engineering & Technology, Manav Rachna International University
Faridabad, India

Abstract: In recent years, Natural Language Processing techniques have been most widely used in various fields like Mining, Information Retrieval, Machine Learning etc. Parsing is one of the important steps when dealing with the processing of natural language. In general, the World Wide Web in which information is present majorly in the form of natural language. This natural language information needs to be dealt properly for tasks like Crawling, Indexing, Ranking etc. Firstly, while parsing the text written in natural language we van follow either the Top-down approach or Bottom-up Approach for validating the text according to the grammar rules specified for it. In this paper, we have given a technique for parsing the natural text following the bottom-up approach. The proposed technique will construct a tree of the natural text by first identifying the fundamental units and then constructing the tree by identifying further higher-order structure entities. It has been found that the proposed technique will produce an efficient parse tree in terms of time complexity.

Keywords: Hierarchical, Parsing, Natural languages Processing, Bottom-Up, Parse-Tree

I. INTRODUCTION

Natural Language Processing (NLP) techniques are very commonly used in several domains of computer science like machine learning, artificial intelligence, linguistic analysis etc. The foremost step of NLP is parsing. Parsing, in general terms is the analysis of given text syntactically. In this the words which are also called lexical items are identified to construct the structure of the text corresponding to the grammatical rules. In common, we have two approaches in Parsing i.e. top-down approach and bottom-up approach. In top-down approach we first start the construction of tree from the starting symbol which is the root of the text. Next, from root of the text various lexical entities are identified which are further used to construct the complete parse tree of the text. In bottom-up approach of parsing the minute level which is having all the basic lexical entities are identified first. Next, from these lexical entities further basic elements are identified which will form the nodes of the tree at different levels. From each level, the bottom-up approach will construct the different novel nodes as per grammatical rules till the root of the text has been reached. In both these approaches the processing is done either directional or non-directional. In non-directional processing approach the parsing is done by retrieving the input which must not be specific in any order. The basic requirement of this non-directional approach is that the whole text/input which is to be parsed needs to be present in the memory. However, the other way approach i.e. directional parsing the processing of construction of parse tree is done in an organized manner and it can be accomplished before the last symbol is examined.

Whenever, the Parsing is done, it means the given text/input is being analyzed syntactically. This will identify that whether the given input/text is according to the specified grammar rules. From computational point of view, parser is one of the components in an interpreter or compiler, which authorize for precise syntax and constructs a data organization implied in the given input lexicalized items. The parser frequently uses a distinct lexical analyzer to construct tokens from the structured characters of input. Most of the common search approaches used in parsing have exponential time dependence in the worst instance. It needs to be noted that the best parsing approach in combination of the above methods involves cubic time. Thus, it becomes deceptive to appear into other approaches specially the linear time common parsing approaches. Considering in view, the time dependency of the static analysis of given input/text common methods like LL and LR parsing techniques following top-down and bottom-up approach respectively are very useful.

In LL; the initial L stands for Left-to right, the other for “recognizing the Left-most construction”. LL parsing, specifically LL(1) is very prevalent. LL(1) parsers are frequently produced by a parser creator but a modest variant can be created by using recursive-descent techniques. Frequently, the LL(1) parsing is used beginning from the last token of the input; it is formerly known as RR(1). There are numerous linear bottom-up approaches; the most influential is LR, where L stand for Left-to-right and the R stand for “recognizing the Right-most construction”. This paper discusses the parsing techniques and gives a novel hierarchical based approach following bottom-up approach in constructing the parse tree.

In this paper section II gives the related work, section III gives the proposed approach. The observation of the proposed approach is given in section IV. Finally, the conclusion is given in section VI.

II. RELATED WORK

LR parser follows bottom-up parsing approach as it stabs to infer the topmost level grammar constructions by developing up from the leaf nodes of the tree which is constructed. Generally, the parser which is denoted by LR(k) is used where the user wants to specify in the approach the value of k, which refers to count of unexamined symbols of input called as "look ahead". These look ahead input symbols are used in building parsing resolutions. It is also known that when the look ahead symbol k equals to one, it is denoted by simply LR. This bottom up approach based parsing has various applications due to its efficiency in terms of time or space. Several programming languages are parsed by means of various variations in LR parser. The most common language used to implement LR is C++, which scans the input string from left to the right generating various tokens called lexical items. The grammar which exists for parsing the string using LR(K) is called context free grammar LR(K).

[1] have given the extensions to the existing dependency parsing algorithm by enhancing it so that it can be applied to rich set of languages. The authors replaced the initial transition-based parses by lookup-based representations. This representations are built from the input words based on LSTMs.

[2] have given a tool known as MaltOptimizer. This tool is designed and developed to ease optimization of Maltparsers. MaltOptimizer achieves an exploration of the data sets and guides the handler by means of three-phase optimization process. This can also be recycled to achieve entirely automatic optimization. It has been shown and proven empirically that the given parser produce results which are more accuarte up to 9 percent.

[4] have given an efficient algorithm which follows dependency parsing with association with deterministic aim. The given algorithm that constructs simple and novel arcs using non-directional way in the structure. According to authors, earlier parsing algorithms following the deterministic aid are dependent on shift-reduce framework. These algorithms examine the input sentence from left-to-right and, at every step of execution. The performance depends on the construction of tree by applying the possible set of rules/actions. The disadvantages of such algorithms are that these follow and evaluate the given input locally, however the decisions on construction is dependent on complex input structures. In comparisons, the authors gave algorithm constructs a dependency tree by selection of best neighbors to further establish the connection between earlier developed nodes recursively. This approach permits unification of structures from previously constructed structures. The results of authors deterministic algorithm is, best-first, O(nlogn), which is precisely and more exact than transition dependent parsers.

[5] discusses the prediction for building systems which are more intelligent and reliable using machine learning techniques. The prediction is significant that is to be applied on sequences of inputs/actions as stochastic processes. The authors have given a prediction algorithm named Active LeZi based on approach of Information Theoretic and algorithms of data compression. The efficiency of the given algorithm in a evaluated by retaining the Active LeZi algorithm to envisage device convention in the home. The synthetic data sets are used to analyze the algorithm performance interfaces between a Smart Home besides the inhabitant.

[6] validate parsing following the dependency approach as penetrating for maximum spanning trees (MSTs) constructed for directed graphs. The illustration is protracted also to parsing which is non-projective by means of Chu-Liu-Edmonds. The authors evaluated the above approaches on Prague Dependency Treebank by means of techniques of machine learning [3] [8] and demonstrated the MST parsing rises competence and exactness for then the non-projective dependencies Parsing.

[7] have given MaltParser, which is a generated for parsing the input using dependency parsing. This data driven parser is given a treebank as an input in dependency format. The given parser can be used to construct a parser based on treebank language. It supports numerous parsing and learning algorithms and permits user-defined feature models. The given parser is freely accessible for research and instructive purposes.

[9] have given a framework for parsing the sentence its Abstract Meaning Representation (AMR) majorly in two stages. In first stage, the dependency parser is used to create a dependency tree for the given input. In the second stage, the authors designed an algorithm based on transition technique which converts the constructed dependency tree to AMR graph. The advantages of the given approach is that dependency parser can be skilled on a much larger data set giving overall accuracy in terms F-measure. In addition to this the rules designed are linguistically instinctive and detect the uniformities during the mapping of constructed dependency tree and the AMR.

[10] presents a novel algorithm for parsing using scalable scene. The algorithm is based on retrieval of image and superpixel matching. The authors focused on classes of rare object, which are significant to achieve more understanding on semantic associations. The major contributions given by the authors are rare class expansion and semantic association description. Firstly, the authors considered the labeled distribution for evaluating the set retrieved by rare class exemplars. Secondly, the global and local association of semantic information with respect to its context is used to improve image retrieval and superpixel matching.

Although many researchers have given various approaches for parsing the input given by the various users, but still there is a need to improve the technique in terms of efficiency considering the time and space from computational point of view. In next section, we are giving the proposed algorithm following the bottom up approach in hierarchical manner.

III. PROPOSED FLOWCHART OF HIERARCHICAL LR ALGORITHM

Bottom-up parsing would be examining a sentence by recognizing words initially. Next, by means of properties of the examined words the grammatical connections are inferred and various phrase structures are constructed to further create a parse tree of the given input sentence. In general, a parser constitutes

i. an input buffer, holding the input string

ii. stack on which to accumulate the variables in terms of terminals and non-terminals

iii. a parsing table which articulates the various grammar rule which are to be applied for given input symbols examined on the top of the constructed stack

In the proposed technique, the parser applies the rule found in the table by corresponding the top-most symbol on the stack with the present symbol in the input stream (column).Initially, the stack already contains two symbols:[S, $],

where '$' is a special terminal to indicate the bottom of the stack and the end of the input stream, and 'S' is the start symbol of the grammar. The parser will attempt to rewrite the contents of this stack to what it sees on the input stream.

The proposed technique is given in figure 1 following the hierarchical approach. The algorithm for the proposed approach is given in Algorithm 1.

```
Algorithm 1:
Input: Set of Rules R {$R_1$, $R_2$, ….$R_i$} , Input String S
Output: Parse Tree
Method:
  1. Add $ as end marker in input string S
  2. Construct State Set $S_S$ and Parse Set $P_S$.
       ¥ {$Var_i$ € S} Extract{$NT_i$ © $T_i$}
  3. Read S Var by Var and ¥ Var
       If($Var_i$€$S_S$) Set State=NR i.e. Not Reduced
       Else
       ($Var_i$€$P_S$) Set State=R i.e. Reduced
         ¥ (Var and $R_i$) apply £ $R_i$ © R
  4. If $ is achieved then Stop
       Else Goto Step 3 and repeat.
       string is accepted.
```

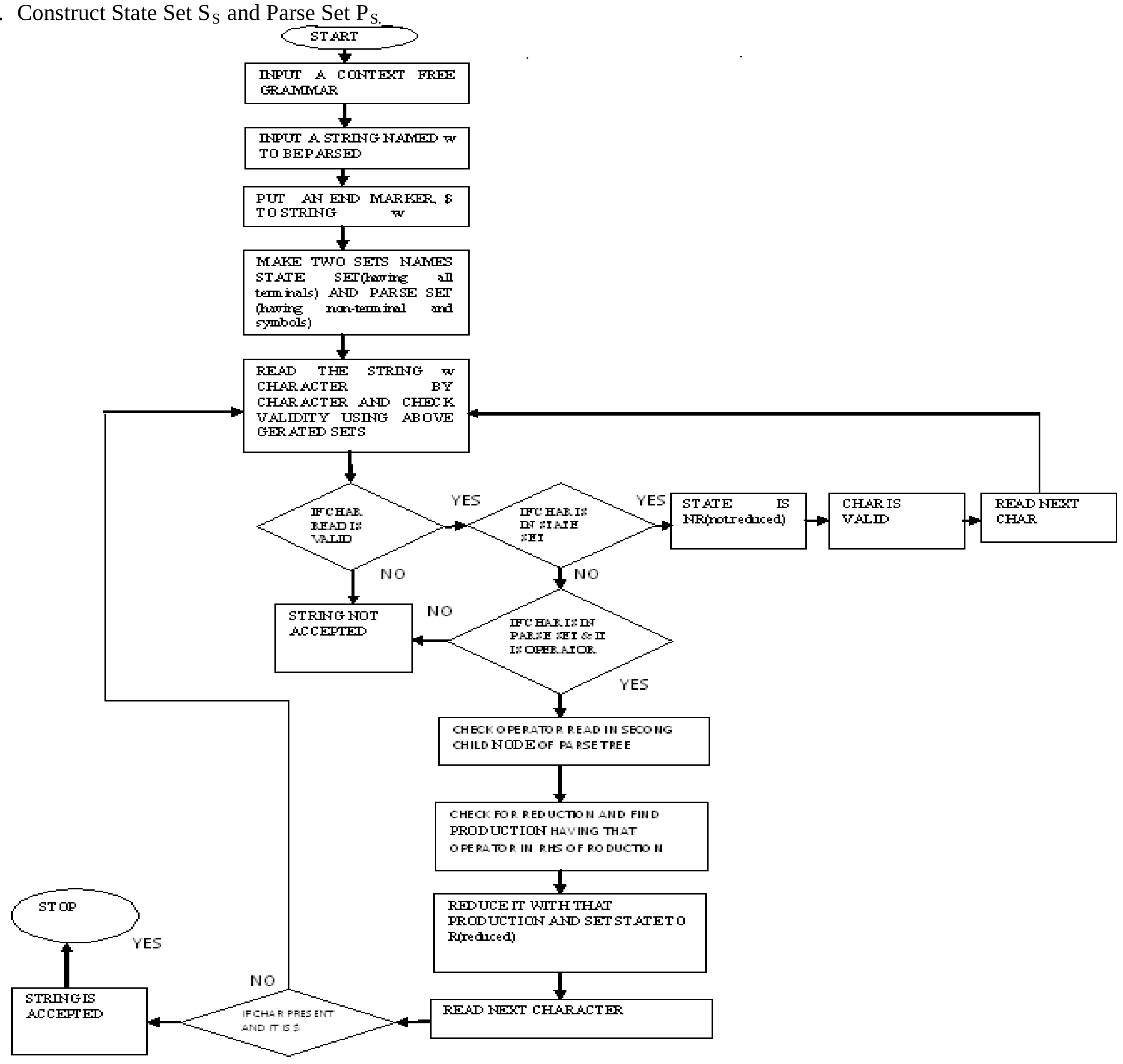
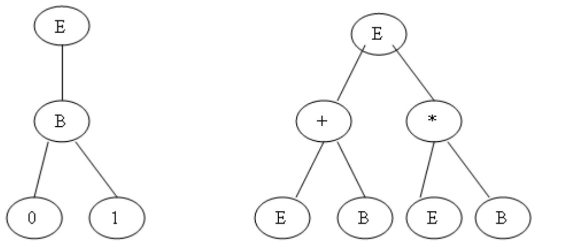

START
INPUT A CONTEXT FREE GRAMMAR
INPUT A STRING NAMED w TO BE PARSED
PUT AN END MARKER, $ TO STRING w
MAKE TWO SETS NAMES STATE SET(having all terminals) AND PARSE SET (having non-terminal and symbols)
READ THE STRING w CHARACTER BY CHARACTER AND CHECK VALIDITY USING ABOVE GERATED SETS
IF CHAR READ IS VALID
YES
IF CHAR IS IN STATE SET
YES
STATE IS NR(not reduced)
CHAR IS VALID
READ NEXT CHAR
NO
STRING NOT ACCEPTED
NO
NO
IF CHAR IS IN PARSE SET & IT IS OPERATOR
YES
CHECK OPERATOR READ IN SECOND CHILD NODE OF PARSE TREE
CHECK FOR REDUCTION AND FIND PRODUCTION HAVING THAT OPERATOR IN RHS OF PRODUCTION
REDUCE IT WITH THAT PRODUCTION AND SET STATE TO R(reduced)
READ NEXT CHARACTER
IF CHAR PRESENT AND IT IS $
NO
YES
STRING IS ACCEPTED
STOP

Figure 1: Overall flow of Proposed Hierarchical Approach

Let us take the same grammar as was taken in previous chapter for LR parsing:
E->E+B
E->E*B
E->B
B->0
B->1
And input string to be parsed is 1+1*1.
For this grammar parse set={E,B,+,*} and state set={0,1}.



Step1: Read 1 present in state set [v | NR]. Read next.
Step2: 1+ (present in parse set) [V | NR]. Read next.

Step3: 1+1 (1 present in state set) [V | NR]. Read parsing tree

Step4: Check + in second child node of parsing tree and reduce the string

1<-E + 1<-B

Now again reduce it to E using rule1.

Step5: E [V | R]. read next character * (present in parse set) i.e. E* [V| NR] again read next.

Step6: E*1 (1 present in state set) [V | NR].Read parsing tree

Step7: Check * in second child node of parsing tree and reduce 1 to B. Now again reduce E*B to E according to rule2 read next.

Step8: Next character not present. E is found then

IV. CONCLUSION

The technique for parsing proposed in this paper has been used to examine whether the specified mathematical expression is rendering to specified rules of grammar or not. The grammar rules are constructed in such a manner that these rules can parse the mathematical expressions which is given as an input. In comparison to LR parsing algorithm in which a parsing table is constructed, this paper gives a novel and efficient technique of parsing following bottom-up approach in which no table is constructed. The proposed technique uses the tree structure which is created step by step as and when the input is parsed which gives the efficiency in terms of computation performed.

One of the simplifications is that the proposed flowchart works only for mathematical expressions. Thus, it supports for +, -, /, %. In conclusion, this paper has given an approach for bottom-up parsing in which string is parsed without construction of item sets and parsing ACTION, GOTO table.

REFERENCES

[1] Ballesteros, M., Dyer, C., & Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with LSTMs. arXiv preprint arXiv:1508.00657.

[2] Ballesteros, M., & Nivre, J. (2012, April). MaltOptimizer: an optimization tool for MaltParser. In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 58-62). Association for Computational Linguistics.

[3] Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research, 3(Jan), 951-991.

[4] Goldberg, Y., & Elhadad, M. (2010, June). An efficient algorithm for easy-first non-directional dependency parsing. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (pp. 742-750). Association for Computational Linguistics.

[5] Gopalratnam, K., & Cook, D. J. (2004). Active lezi: An incremental parsing algorithm for sequential prediction. International Journal on Artificial Intelligence Tools, 13(04), 917-929.

[6] McDonald, R., Pereira, F., Ribarov, K., & Hajič, J. (2005, October). Non-projective dependency parsing using spanning tree algorithms. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (pp. 523-530). Association for Computational Linguistics.

[7] Nivre, J., Hall, J., & Nilsson, J. (2006, May). Maltparser: A data-driven parser-generator for dependency parsing. In Proceedings of LREC (Vol. 6, pp. 2216-2219).

[8] Polman, C. H., Reingold, S. C., Edan, G., Filippi, M., Hartung, H. P., Kappos, L., ... & Sandberg-Wollheim, M. (2005). Diagnostic criteria for multiple sclerosis: 2005 revisions to the "McDonald Criteria". Annals of neurology, 58(6), 840-846.

[9] Wang, C., Xue, N., Pradhan, S., & Pradhan, S. (2015). A Transition-based Algorithm for AMR Parsing. In HLT-NAACL (pp. 366-375).

[10] Yang, J., Price, B., Cohen, S., & Yang, M. H. (2014). Context driven scene parsing with attention to rare classes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3294-3301).