Fuzzy LALR Parser for Parsing Natural Language Sentences of English Language

Suvarna G Kanakaraddi[1], Suvarna S Nandyal[2]

[1]*BVB College of Engineering & Technology, Hubli-580031, Karnataka, India*
[2]*PDA College of Engineering, Kalaburgi-585102, Karnataka, India*

***Abstract*—The Natural Language Processing (NLP) includes scope of computational methods for examining and speaking to actually happening writings at least one levels of semantic investigation with the end goal of accomplishing human-like dialect preparing for a scope of assignments or applications. For performing sentence structure investigation, the Fuzzy LALR (FLALR) parser is the best-known and most proficient parsing instrument. Really, the progressions of the setting free preparations are required to outline the well-working FLALR parser. In this paper FLALR parser, is presented, and its application to common dialect parsing is talked about. A FLALR parser is a move diminish parser which is deterministically guided by a parsing table. A parsing table can be acquired consequently from a setting free expression structure linguistic use. FLALR parsers can't oversee vague sentence structures, for example, common dialect syntaxes, on the grounds that their parsing tables would have increase characterized sections, which block deterministic parsing. FLALR parser, be that as it may, can deal with duplicate characterized passages, utilizing a dynamic programming strategy. At the point when an input sentence is ambiguous, the parser delivers all conceivable parse trees without parsing any piece of the information sentence more than once similarly.**

***Index Terms*—NLP, LR, FLALR, FCLR, FSLR and Fuzzy Context free Grammar.**

I. INTRODUCTION

Language is one of the key parts of human conduct and is vital segment of our lives. In composed frame it fills in as a long –term record of learning starting with one era then onto the next. In talked shape it fills in as our essential methods for organizing our everyday conduct with others. Characteristic Language Processing is a hypothetically persuaded scope of computational systems for breaking down and speaking to actually happening writings at least one levels of semantic examination with the end goal of accomplishing human-like dialect preparing for a scope of errands or applications.

The FLALR (Look Ahead-LR) parsing strategy is between Fuzzy Simple LR (FSLR) and Fuzzy Canonical LR (FCLR) both as far as energy of parsing linguistic uses and simplicity of usage. This technique is regularly utilized as a part of training in light of the fact that the tables gotten by it are extensively littler than the FCLR tables, yet most regular syntactic builds of programming dialects can be communicated advantageously by a FLALR sentence structure. FLALR parsers have been created initially to programme dialect of compilers. A FLALR parser is a look ahead parser which is deterministically guided by a parsing table showing what move ought to be made next. The parsing table can be acquired consequently from a setting free expression structure linguistic use, utilizing a calculation. The LR parsers have occasionally been utilized for Natural Language Processing likely in light of the fact that:

1. It has been believed that characteristic dialects are not setting free, while FLALR parsers can bargain just with setting free dialects.
2. Characteristic dialects are uncertain, while standard FLALR parsers can't deal with vague dialects.

The current writing demonstrates that the conviction “natural languages are not context-free" is not necessarily true and there is no explanation behind us to surrender the setting flexibility of common dialects. Our fundamental concern is the means by which to adapt to the uncertainty of regular dialects, and this worry is tended to in the accompanying segments.

Interestingly with Aho et al [1]. Creators approach is to augment LR parsers, with the goal that they can deal with different passages and deliver more than one parse tree if necessary. Here LR parsers are upgraded by utilizing Fuzzy rationale. Next Section talks about the study led.

II. LITERATURE SURVEY

This section discusses about the survey about the research conducted by various authors. Many authors have proposed different techniques for processing natural language.

Nilsson et al. [2] have proposed novel approach to extract structural information from source code using state-of-the-art parser technologies for natural languages. Here natural language parsing techniques are applied for information extraction from formally structured information sources, such as programs. This method automatically generates the language specific information extractor using machine learning and training of a generic parsing approach. The training data can be generated automatically.

Chen et al. [3] proposed a parsing model which is factored into a lexical and a constituent model, which enables interaction between tagging and parsing. Experimental result achieves statistically significant improvement in both parsing and tagging accuracy on both English and Chinese.

Tomita et al. [4] designed an efficient parsing algorithm for natural language interfaces using context-free grammars. Here LR parsing algorithm is designed, which computes an LR shift-reduce parsing table from a given augmented context-free grammar. The algorithm parses a sentence strictly from left to right on-line, that is, it starts parsing as soon as the user types in the first word of a sentence, without waiting for completion of the sentence.

Liang Chen and Naoyuki Tokuda [5] have developed a table look- up parser for intelligent language tutoring system (ILTS), which is based on the template structure for the answers of questions. It is shown that the number of different grammar structure of sentences in a template is much smaller than that of different correct sentences.

Verd'u-Mas et al. [6] have compared three different approaches of probabilistic context-free grammar for natural language parsing from a tree bank corpus: (1) a model that simply extracts the rules contained in the corpus and counts the number of occurrences of each rule; (2) a model that also stores information about the parent node's category, and (3) a model that estimates the probabilities according to a generalized k-gram scheme for trees with k = 3. Proposed a Probabilistic Context Free Grammar.

Mochamad Vicky Ghani Aziz et al.[7] proposed natural language processing approach the syntax and semantic analysis to improve the structure of words and sentence parsing so that it can classify traffic conditions tweet of the sentence.

Andrew Begel et al.[8] have developed a combined lexer and parser generator which enables many classes of embedded languages and ambiguities in spoken language. Enhanced lexing and parsing algorithms in harmonia framework to analyze lexical, syntactic and semantic ambiguities.

Mulik, et al. [9] have compared three parsing methods like Conventional, Fuzzy and NLP. They have analyzed that parsers using NLP techniques have major advantage over classical and fuzzy parsing.

Yuncheng Jiang , Yong Tang [10] have developed a computing with words, a formal interval type-2 fyzzy model . Which combines interval type-2 fuzzy set Pushdown automata theory and automaton theory, as a computational model of computing with words. Build the extension principles to extend from computing with values to computing with words.

III. FLALR PARSER

The FLALR parsing table development calculation is precisely the same as the calculation for LR parsers. Just the distinction is that a LR parsing table may have various passages. "s" in the activity table (the left piece of the table) demonstrates the activity "move single word from input support onto the stack, and go to state n". "r" shows the activity "lessen constituents on the stack utilizing guideline n". "acc" remains for the activity "acknowledge". Goto table (the correct piece of the table) chooses to what express the parser ought to pursue a diminish activity.

Once a parsing table has numerous passages, deterministic parsing is not any more conceivable; some sort of non determinism is fundamental. Dynamic programming approach, which is portrayed beneath, is a great deal more productive than traditional approach and, makes FLALR parsing practical. At the point when a procedure experiences a numerous passage with n distinctive activities, the procedure is part into n procedures, and they are executed independently and parallelly. Each procedure is proceeded until either a "blunder" or an "acknowledge" activity is found. The procedures are, nonetheless, synchronized in the accompanying way: When a procedure "moves" a word, it holds up until every different procedure "move" the word. Naturally, all procedures dependably take a gander at a similar word. After all procedures move a word, the framework may locate that at least two procedures are in a similar express; that is, a few procedures have a typical state number on the highest point of their stacks. These procedures would do the precisely same thing until the point when that regular state number is flown from their stacks by some "decrease" activity. In creators approach, this normal part is prepared just once. When at least two procedures in a typical state are discovered, they are consolidated into one process. This consolidating component ensures that any piece of an information sentence is parsed close to once in a similar way." This makes the parsing considerably more effective.

With FLALR (lookahead LR) parsing, it endeavor to decrease the quantity of states in a LR (1) parser by consolidating comparative states. This decreases the quantity of states to the same as FSLR (1), yet at the same time holds a portion of the energy of the LR (1) lookaheads.

FLALR is a software engineering acronym for look ahead left to right. It is a strategy for parsing coding languages or unstructured content documents. Parsing is perceiving designs in input explanations that match leads in a language structure. While parsing a coding, a FLALR parser:

- Uses a LOOK AHEAD symbol to aid the recognition process,
- Reads input statements from LEFT to right,
- Makes reductions on the RIGHT first, working backward toward the left side of the grammar.

A. FLALR (1) Grammars

A formal meaning of what makes a syntax FLALR(1) can't be effortlessly exemplified in an arrangement of principles, since it needs to look past the particulars of a creation in separation to consider alternate circumstances where the generation shows up on the highest point of the stack and what happens when we blend those circumstances. Rather we express that what makes a linguistic use FLALR (1) is the nonappearance of contentions in its parser. On the off chance that you assemble the parser and it is without strife, it suggests the language structure is FLALR (1) and the other way around. FLALR (1) is a subset of LR(1) and a superset of F SLR(1). A language structure that is not LR(1) is unquestionably not FLALR(1), since whatever contention happened in the first LR(1) parser will even now be available in the FLALR(1). A syntax that is LR (1) could possibly be FLALR (1) contingent upon whether consolidating presents clashes. A linguistic use that is F SLR (1) is certainly FLALR (1). A linguistic use that is not FSLR (1) could possibly be FLALR(1) contingent upon whether the more exact lookaheads resolve the FSLR(1) clashes. FLALR (1) has ended up being the most utilized variation of the LR family. The shortcoming of the FSLR (1) and LR(0) parsers mean they are just equipped for taking care of a little arrangement of syntaxes. The broad memory needs of LR (1) made it mull for quite a long while as a hypothetically fascinating yet unmanageable approach. It was the approach of FLALR (1) that offered a decent harmony between the energy of the particular lookaheads and table size.

B. Construction Idea

- Construct the set of LR (1) items.
- Merge the sets with common core together as one set, if no conflict (shift-shift or shift-reduce) arises.
- If a conflict arises it implies that the grammar is not FLALR.
- The parsing table is constructed from the collection of merged sets of items using the same algorithm for LR (1) parsing.

IV. METHODOLOGY

To compute the LR (1) configurating sets initially implies we won't spare whenever or exertion in building a FLALR parser. At the point when the parser is executing, it can work with the compacted table, in this way sparing memory. The distinction can be a request of extent in the quantity of states. However there is a more effective procedure for building the FLALR (1) states called well ordered combining. The thought is that you blend the configurating sets as you go, as opposed to holding up until the end to locate the indistinguishable ones. Sets of states are developed as in the LR (1) technique, however at each point where another set is brought forth, first verify whether it might be converged with a current set. This implies looking at alternate states to check whether one with a similar center as of now exists. Assuming this is the case, consolidate the new set with the current one, generally include it typically.

A .Construction Idea

Fuzzy Context-Free Grammar (FCFG), as an extension of context-free grammar , has been introduced to express uncertainty, ambiguity, and vagueness in natural language fragments. Here production rules are designed by considering noun phrase (NP), verb phrase (VP), preposition (PP), adjective (ADJP) [11] and fuzzy membership values are assigned for each rule. Membership values are assigned randomly to each of these rules, finally the values for the entire set of rules for each phrase sums up to 1.

Consider commonly used Production rules for construction of English language sentences are as follows [12],

1) S→NP VP (1.0)	12) VP→ v VP (0.1)
2) S→ aux NP VP (1.0)	13) VP→ v NP VP (0.2)
3) NP→art n (0.2)	14) VP→ v ADJP (0.1)
4) NP→pron (0.2)	15) VP→ TO VP (0.2)
5) NP→ n (0.1)	16) VP→ v NP PP (0.1)
6) NP→ NP PP (0.2)	17) VP→ v PP (0.1)
7) NP→ propn (0.1)	18) PP→prep NP (1.0)
8) NP→ NOM (0.2)	19) ADJP→adj (0.5)
9) NOM→adj n (1.0)	20) ADJP→adj VP (0.5)
10) VP→v (0.1)	21) TO→ to (1.0)
11) VP→v NP (0.1)	

These production rules are used further for computing f FIRST, FOLLOW and closure of item sets. Further using these methods FLALR algorithm is developed.

B. Computation of FIRST

To compute FIRST(X) for all grammar symbols X, apply the following rules until no more terminals or Є can be added to any FIRST set [12].

1. If X is a terminal then FIRST (X) = {X}
2. If X is a non terminal and X→ Y_1Y_2……Y_k is a production for some k ≥ 1, then place a in FIRST (X) if for some I, a is in FIRST(Y_i), and Є is in all of FIRST(Y_1)……..FIRST(Y_{i-1}); that is $Y_1 - Y_{i-1} \overset{*}{\Longrightarrow}$ Є. If Є is in FIRST (Y_j) for all j= 1,2,…….k, then add Є to FIRST(X) .
3. If X→ Є is a production then add Є to FIRST(X).[4]

C. Computation of FOLLOW

To compute FOLLOW (A) for all non terminals A, apply the following rules until nothing can be added to FOLLOW set.

1. Place $ in FOLLOW (S) , where S is the start symbol and $ is the input right end marker.
2. IF there is a production A→ α B β , then everything in FIRST(β) except Є is in FOLLOW(B).
3. If there is a production A→ α B , or a production A→ α B β, where FIRST(β) contains Є , then everything in FOLLOW(A) is in FOLLOW(B) [4]

D .Computation of closure

Set of Items clousure (I)

{

J = I;

repeat

for (each item $A \xrightarrow{\lambda i} \alpha .B \beta$ in J) where

$^{\lambda}i = [0..1]$

for (each production B→ γ of G)

if ($B \xrightarrow{\lambda i} .\gamma$ is not in J)

add $B \xrightarrow{\lambda i} .. \gamma$ to J ;

Until no more items are added to J on one round :

Return J ; }[4]

In the closure module, the parameters to be passed to the closure computation are the state number of the new state to be formed and the production count. A dot character is used to keep track of the productions processed. The character next to the dot is checked for its non-terminal. If yes, then all the productions starting with that non-terminal are added to the new state.

V. FLALR ALGORITHM

The following algorithm is used for construction of FLALR parsing table. In this algorithm the similar states from FCLR parser are merged in order to minimize the number of states.

FLALR parsing table construction

INPUT: An augmented grammar G'[4].

OUTPUT: The FLALR Parsing table functions ACTION and GOTO for G'.

METHOD:

1. Construct C = (I0,I1,……..In) the collection of sets of LR(1) Items for G' .
2. For each core present among the set of LR(1) items, find all sets having that core, and replace these sets by their union.
3. Let C' = {J0, J1, …..Jm} be the resulting sets of LR (1) items. The parsing actions for state i are constructed from Ji in the same manner as in Canonical LR algorithm. If there is a parsing action conflict, the algorithm fails to produce a parser, and the grammar is said not to be LALR (1).
4. The GOTO table is constructed as follows. If J is the union of one or more sets of LR(1) items, that is, J = I1 ∩ I2 ∩ ……∩ Ik , then the cores of GOTO (I1,X), GOTO(I2, X), ……..,GOTO(Ik, X) are the same, since I1,I2,….,Ik all have the same core. Let K be the union of all sets of items having the same core as GOTO (I1,X). Then GOTO (J,X) = K.

A. FLALR item sets

Item sets construction for LALR is computed using Fuzzy context free grammar, FIRST and FOLLOW computed for the grammar considered. Here Item sets are constructed by using Canonical item sets. Similar item sets are merged here to minimize the item sets count compare to Canonical Item sets.

Goto

GOTO (I,X) where I is a set of items and X is a grammar symbol. GOTO (I,X) is defined to be the closure of the set of all items $[A \xrightarrow{\lambda i} \alpha X.\beta]$ such that $[A \xrightarrow{\lambda i} \alpha.X\beta]$ is in I. The GOTO function is used to define transitions for a grammar, GOTO (I, X) specifies the transition from state I under input X.

In the goto module, first on the given input grammar symbol X, check whether if the new state being created has the same kernel items. If the kernel items are same, then a new state is not created and a transition on the given input grammar symbol is pointed back to the same state. If the kernel items are not same then a new state is created, the closure is calculated, thus determining the items of the new state.

B. LALR - Action and GOTO Table

Action and Goto table for LALR is constructed from First, Follow and the computed item sets. Using this table English sentence input is parsed to check whether the given input English sentence is syntactically correct or not.

VI. RESULTS

The LALR parser makes use of item sets to determine its actions. The item sets are generated by the computation of closure and goto. Initially the computation of closure and goto is computed using the algorithm, then the action table consisting of shift/reduce actions is generated making use of the results of closure and goto. Finally the max-min is computed for the parsed result. Following Fig.1 shows the

input sentence given to the parser and after parsing the completely parsed sentence with its associated fuzzy value.

Input sentence and completely parsed result.

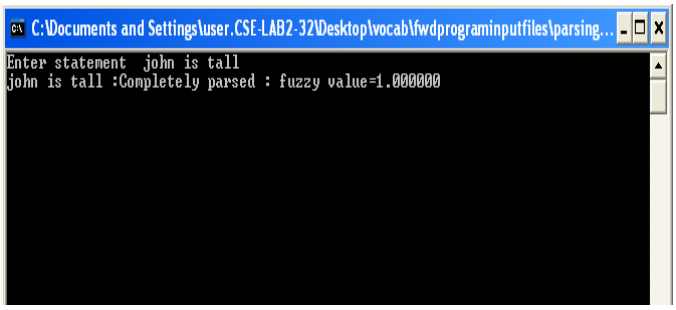

Fig . 1 Input for FLALR and output

Following figure Fig. 2 shows the Permutations generated and parsing status. Here result shows the partially parsed status of the sentence with its associated fuzzy value. Completely parsed sentence is also shown with its associated fuzzy value. Completely parsed sentence will have high fuzzy value compare to incompletely parsed sentences. The degree of fuzzy value varies with the number of words parsed. In a sentence number of words parsed will represent the syntactic correctness of the words and remaining unparsed words are not syntactically correct. Which shows that even incompletely parsed sentences will shows the syntactic correctness and its degree of fuzziness.

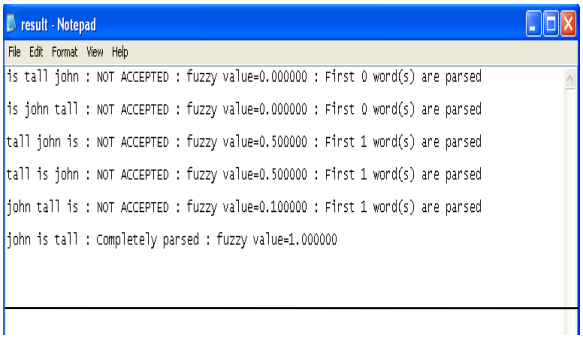

Fig .2 Permutations and output

VII. Conclusions

In this paper author has developed a parsing technique called Fuzzy LALR (FLALR), Here Fuzzy context free grammar is designed for parsing Natural language. FLALR is implemented in 'C' Programming Language. Considering English language sentence as an input, permutations are generated and for the generated permutations Fuzzy LALR algorithm is applied. Finally Fuzzy max-min technique is applied to get the degree of fuzziness. Experimental results have been presented here. Our research work involves the design of fuzzy parsing algorithms and implementation to provide the better results compare to conventional approach. The main advantage of fuzzy parsers over conventional parser is that it gives degree of fuzziness and syntactic correctness for partially parsed sentences but in conventional parsers the sentences parsed completely are only accepted and rejected completely if it is partially parsed. Syntax analysis helps to improve recognition rates significantly. Author concludes that one of the major advantage of this method is , compare to Fuzzy Simple LR and Fuzzy Canonical LR , number of states generated are less in Fuzzy LALR. Which shows that Fuzzy LALR provides better result compare to Fuzzy Simple LR and Fuzzy Canonical LR.

Acknowledgment

We are thankful to our organization for providing an infrastructure in conducting this research. We express our gratitude to Dr. Ashok Shetter, Dr. P.G.Tewari , Dr P S Hiremath and HOD Dr .G.H Joshi for their continuous support and encouragement.

References

[1] Alfred V Aho, "Compilers Principles, Techniques, and Tools", Pearson Education, pp.191-217, 2006.

[2] Jens Nilson, Welf Lowe, Johan Hall , Joakim Nivre, "Natural Language parsing for fact exatraction from source code, ICPC-IEEE 2009.

[3] Xian chen and Chunyu kit, "Improving parts- of -speech tagging for context-free parsing", Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 1260–1268, Chiang Mai, Thailand, November 8 – 13, 2011.

[4] Masaru Tomita, "An efficient augmented context free parsing algorithm", Computational Linguistics, Volume 13, Numbers 1-2, January-June 1987.

[5] Liang Chen, Naoyuki Tokunda, "A special Parser for Learning English Composition Error Analysis & Learners' Model for ILTS.

[6] Jose L. Verd´u-Mas, Mikel L. Forcada, Rafael C. Carrasco, and Jorge Calera-Rubio, "Tree *k*-Grammar Models for Natural Language Modelling and Parsing", SSPR&SPR 2002, LNCS 2396, pp. 56–63, Springer-Verlag Berlin Heidelberg 2002.

[7] Mochamad Vicky Ghani Aziz , Ary Setijadi Prihatmanto , Diotra Henriyan , Rifki WLjaya, "Design and Implementation of Natural Language Processing with Syntax and Semantic Analysis for Extract Traffic Conditions from Social Media Data", 2015 IEEE 5th International Conference on System Engineering and Technology, Aug. 10 - 11, UiTM, Shah Alam, Malaysia , 978-1-4673-6713-4/15/$31.00 ©2015 IEEE

[8] Andrew Begel, Susan L. Graham, "Language Analysis and Tools for Ambiguous 1 Input Streams", Electronic Notes in Theoretical Computer Science 110 (2004) 75–96

[9] Sunanda Mulik, Sheetal Shinde, Smita Kapase ," Comparison of Parsing Techniques for Formal Languages", International Journal on Computer Science and Engineering (IJCSE) ISSN: 0975-3397 , VOL 3 No 4 Apr 2011.

[10] Yuncheng Jiang , Yong Tang, "An interval type-2 fuzzy model of computing with words",Information Sciences 281 (2014) 418–442

[11] Erkki Luuk, "The noun/verb and predicate/argument structures", Lingua 119 ,Elsevier Publication , pp 1707–1727, 2009.

[12] Suvarna G Kanakaraddi,V Ramaswamy," Natural Language Parsing using Fuzzy Simple LR (FSLR) Parser", 978-1-4799-2572-8/14/$31.00_c 2014 IEEE.

AUTHORS' PROFILES

Suvarna G Kanakaraddi : Bachelor degree in computer science and Masters degree in Computer science from VTU Belgaum,Karnataka, india. Pursuing research in Artificial Intelligence in VTU Belgaum. Working as Associate professor in Computer Science and Engineering. Areas of interest include Data mining, Cloud Computing , Storage technology. Computer Networks. Working as SPOC for EMC Bangalore.

Suvarna S Nandyal : Bachelor degree in computer science and Masters degree in Computer science from VTU Belgaum,Karnataka, India. Completed Ph.D in Image Processing domain , from JNTU Hyderabad. Working as Professor and Head in Computer Science and Engineering and Research center Head for Computer Science & Engineering. Areas of interest include Image Processing, Data mining, Cloud Computing, and Computer Networks.