Bottom up Fuzzy Parsers: Fuzzy Simple LR, Fuzzy Canonical LR and Fuzzy LALR Parsers for Parsing Natural language

Suvarna G Kanakaraddi[1], Suvarna S Nandyal[2]
[1]Dept of Computer Science & Engineering, BVBCET Hubli, Karnataka , India,
[2]Dept of Computer Science & Engineering, PDA College of Engineering, Kalaburgi, Karnataka, India
[1]suvarna_gk@bvb.edu,
[2]suvarna_nandyal@yahoo.com

***Abstract* - Humans convey the information through natural language. It is a prevailing tool used by peoples in daily life. Natural Language Processing (NLP) involves methods for analyzing the words through many levels of linguistic analysis. Language exhibits mainly two functionalities. First functionality stipulates syntax and second functionality specifies semantics of the language. The development of Fuzzy parsers for performing syntax analysis of the Natural Language (NL) is described in this paper. Conventional Bottom up parsing algorithms such as Simple Left to Right (SLR), Canonical Left to Right(CLR) and Look Ahead Left to Right(LALR) parsers are enhanced by applying Fuzzy Logic (FL). Left to Right (LR) syntax analysis technique is a constructive method for parsing context free languages.**

Keywords: FSLR, FCALR, FLALR, LR, NL, NLP

I. INTRODUCTION

Natural language understanding (NLU) is a part of Natural Language Processing. NLU is an elementary problem in NLP in conditions of its theoretical and pragmatic significance. Currently much improvement is made at many levels of NLP tasks, which provides great opportunity for deeper natural language understanding. Language must perform at least two functions. In first function, it should identify the sentence structure of the language. In second function, it should indicate the semantics of the language. Compilers for the programming language should identify the rules of the language specification. Compilers should translate the given input into an object language program style and it should be consistent with semantic pattern of the language. If the input consists of syntax errors, compiler should identify the occurrence of fault and also it should show the location of the error. To perform these functions every compiler has a technique within it called a parser.

Fuzzy Context Free Grammar can be used to identify the language rules of a programming language. If the sentence structure is designed cautiously, then much of the semantics of the language will narrate about the rules of the grammar. There are many different types of parsers. In this paper enhanced bottom up parser is described. These parsers are proficient and well suited for use in compilers for programming languages. For huge collection of context free grammars Left to Right(LR) parsers can be automatically generated. Main objective of this approach is to construct LR parsers from certain context-free grammars, even some uncertainties.

A vital characteristic of the parser generation algorithm is the automatic detection of ambiguities and hard to parse constructs in the language pattern. LR parser algorithm is driven by a parser table, a data structure which contains the syntax of the computer language is being parsed. LR parser processes most of the programming languages with the help of parser table. The parser table is constructed by an approach known as parser generator. Authors have enhanced LR parser algorithms by using Fuzzy Context Free Grammar (FCFG).

II. METHODOLOGY

Conventional parsing methods are Enhances by fuzzy constructs. Here English language sentence is given as an input and syntactic correctness is tested using fuzzy bottom up parsers. Fuzzy parser model is represented in Fig.1 .This section describes the construction of fuzzy context free grammar for the parsers.

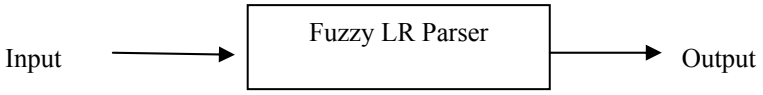

Fig. 1 Fuzzy Parser Model

A. Fuzzy Context Free Grammar (FCFG):

Consider commonly used Production rules for construction of English language sentences are as follows [1],[3]

1) S→NP VP (1.0) 2) S→ aux NP VP (1.0)
3) NP→art n (0.2) 4) NP→pron (0.2)

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

Using these production rules bottom up Left to Right parser tables are constructed. Table holds the information regarding action and go to functions. Actions performed are shift and Reduce. Go to table depicts the state number. In action table shift and reduce functions with its fuzzy membership values are shown.

B. Fuzzy Simple Left to Right (FSLR) Algorithm

FSLR is a Left to Right bottom up parser which uses a follow set to remove conflicts from its action table. It has fewer conflict states than LR (0) parser. The parser construction for FSLR is nearly identical to an LR (0) parser except that generation of reduce actions depends on the follow set.

Algorithm for Construction of FSLR parser table [1]

Consider an augmented grammar G'.

Construct parsing table functions for FSLR such as ACTION and GOTO for the given grammar G'.

METHOD:

1. Construct collection of canonical item sets, C = (I0, I1,……..In) for the grammar G' .
2. State i is build from Ii the actions of parsing for state i are determined as follows,
   a) If [A $\xrightarrow{\lambda i}$ α .a β] is in Ii and GOTO(Ii, a) = Ij, then set ACTION [i,a] to "shift j/membership value" .Here a must be a terminal.
   b) If [A $\xrightarrow{\lambda i}$ α .] is in Ii, then set ACTION [i , a] to "reduce A $\xrightarrow{\lambda i}$ α " for all a in FOLLOW(A); here A may not be S'.
   c) If [S' S.] is in Ii, then set ACTION [i, $] to "accept"
3. The goto transitions of state i are created for all non terminals A by using the rule : If GOTO(Ii,A) = Ij , then GOTO[i, A] = j .
4. Entries which are not defined by rules 2 and 3 are made error.
5. The first state of the parser is constructed using set of items containing [S'→ .S]

FSLR-Action and GO TO Table

FSLR action and Go to table are constructed from First, Follow and the computed item sets. Here the table is constructed from Fuzzy Context Free Grammar. Further this table is utilized to perform syntax analysis of English sentence.

C. Algorithm for construction of Fuzzy Canonical Left to Right Parser table [2]

Augmented grammar G' is given as an input.

Fuzzy Canonical Left to Right Parser table functions ACTION and GOTO are computed for the G'.

METHOD:

1. Collection of canonical item sets P= (I0,I1,……..In) of LR(1) computed for G' .

2. State i is computed using Ii. The parsing actions of state i are designed as follows,

a) If [Q α .a β] is in Ii and GOTO($\xrightarrow{\lambda i}$, a) = Ij, then ACTION [i,a] is set to "shift j/membership value" .Here a must be a terminal.

b) If [Q α .] is in Ii, then ACTION [i , a] is set to "reduce Q α " for all a in FOLLOW(Q); here Q may not be S'. $\xrightarrow{\lambda i}$

c) If [S' S.] is in Ii, then ACTION [i, $] is said to "accept" .

If any contradictory actions occur from the above rules, then the grammar is not LR(1) . The algorithm fails to produce a parser for this case.

3. The goto transitions for state i are determined for all non terminals using the rule : If GOTO (Ii, Q) = Ij , then GOTO [i, Q] = j .

4. Entries not defined by rules 2 and 3 are made error.

5. The initial state of the parser is constructed from set of items containing [S' .S, $]

D. Constructing an FLALR Parsing table [2]

An augmented grammar G' is considered as an input.

FLALR Parsing table functions ACTION and GOTO are computed for G'.

METHOD:

1. Construct collection of item sets S = (P0,P1,……..Pn) 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 S' = {Q0, Q1, …..Qm} be the resulting sets of LR (1) items. The parsing actions for state i are determined from Ji in the same approach 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 an LALR (1).
4. The GOTO table is constructed as follows. If J is the union of one or many sets of LR(1) items, that is, J = P1 ∩ P2 ∩ ……∩ Pk , then the cores of GOTO (P1,X), GOTO(P2, X), ……..,GOTO(Pk, X) are the same, since P1,P2,….,Pk all have the same core. Let K be the union of all sets of items having the same core as GOTO (P1,X). Then GOTO (Q,X) = K .

III. RESULT ANALYSIS

The following results shows the input sentence and permutations generated and also shows the degree of fuzziness for the parsed input. Finally it shows the completely parsed sentence with maximum fuzziness. Fig 2. Shows the input given for the FSLR parser.

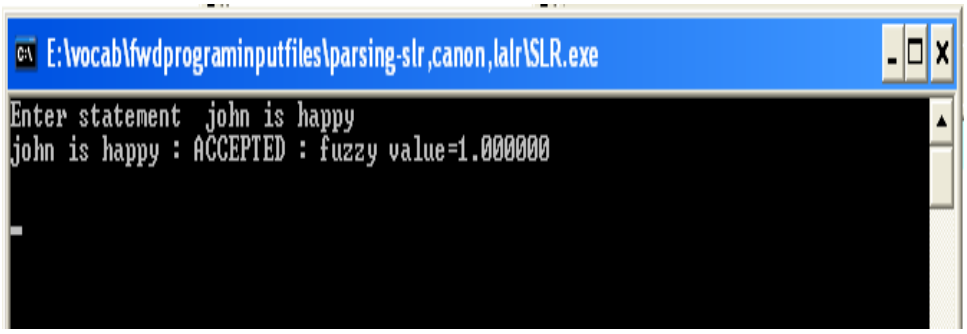

Fig 2. FSLR Input

The following Fig 3. Shows the permutations generated and parsing status showing the degree of fuzziness of the input sentence and also shows the completely parsed sentence.

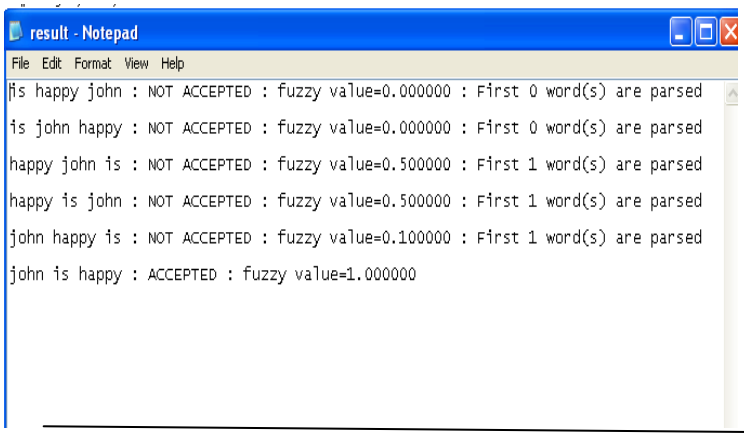

Fig 3. Permutations of FSLR input

Here English sentence input is parsed using Fuzzy Canonical Left to Right (FCLR) parser. In this approach initially the grammar rules are defined and action and go to table is constructed, using this table parsing is done for the given input sentence. Result has been explained in Fig 4. Shows completely parsed sentence with its associated fuzzy value.

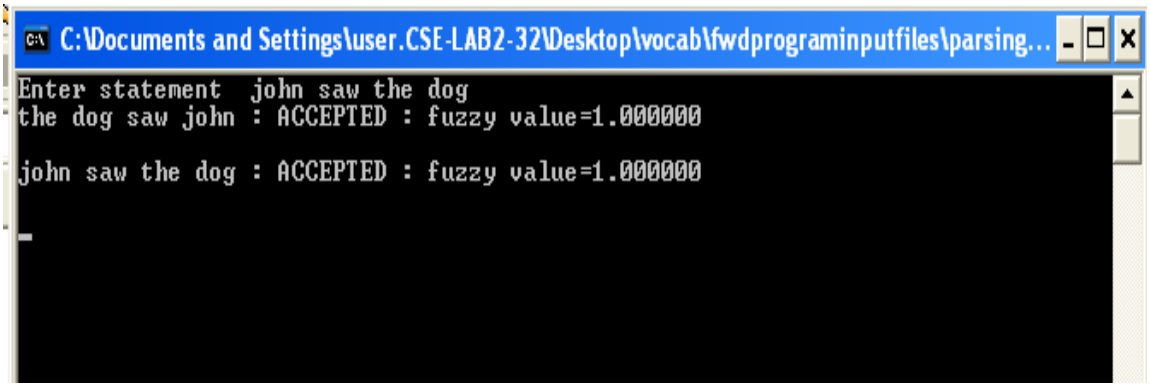

Fig4. Parsing result

Following Fig 5 shows the permutations generated for the input sentence.

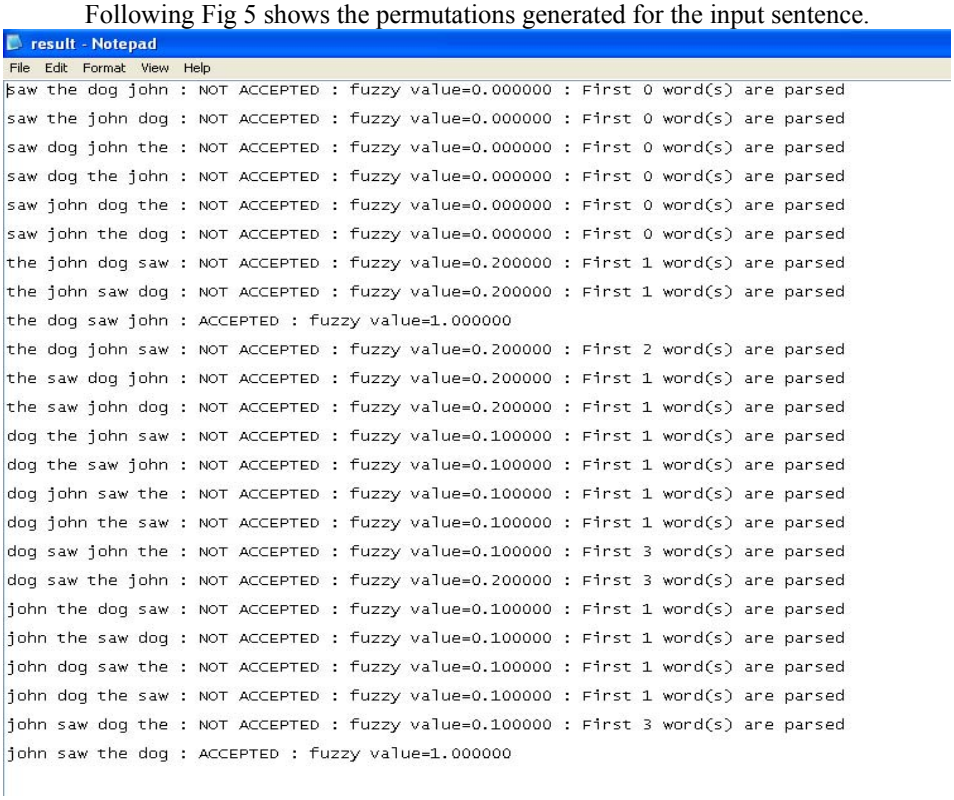

Fig 5 Permutations of FCLR input

Number of states generated in FSLR for the chosen FCFG are 29 and for FCLR are 53. States generated in FCLR are more compare to FSLR. In FLALR the states which are identical in FCLR are merged in one state and the table is created for action and go to. Here FLALR table is having 29 states which is same as FSLR. Number of states generated in both the approaches are same. Among all these approaches FLALR provides better result. Fuzzy max min technique is applied to know the degree of fuzziness of each word.

IV. CONCLUSION

An enhanced parsing technique such as Fuzzy Simple Left to Right (FSLR), Fuzzy Canonical LR (FCLR) And Fuzzy Look Ahead Left to Right (FLALR) parsers are discussed in this paper. Here FCFG is designed for parsing Natural language. These algorithms are implemented in 'C' Programming Language. Considering English language sentence as an input, permutations are generated. For the generated permutations these Fuzzy LR algorithms are applied. Finally Fuzzy max-min technique is applied to get the degree of fuzziness. Results are discussed in this paper. Our research work involves design and implementation 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 assists to get better identification rates significantly. We conclude that among these algorithms FLALR gives improved result and minimizes the states compare to FCLR. Main limitation of FCLR algorithm is number of states generated are more compare to FSLR algorithm.

REFERENCES

[1] Suvarna G kanakaraddi ,V Ramaswamy "Natural Language Parsing using Fuzzy Simple LR (FSLR) Parser", 2014 IEEE International Advance Computing Conference (IACC), Goargao, pp1337-1341
[2] Alfred V Aho, "Compilers Principles, Techniques, and Tools", Pearson Education, pp.191-217.
[3] John N Mordeson, "Fuzzy Automata and Languages", Chapman & Hall/CRC Crc Press company, Washington D.C, pp. 127-137.

AUTHOR PROFILE

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.