3rd International Conference on Science,Technology and Management
India International Center, New Delhi (ICSTM-16)
17 January 2016, www.conferenceworld.in 978-81-932074-0-6

STUDY AND IMPLEMATATION OF STANFORD PARSER

Jyoti Patel[1], Om Prakash Yadav[2]

[1]*M.Tech.Scholar, Dept.of Computer Science & Engg. CSIT, Durg, (India)*

[2]*Associate Professor, Dept.of Electronic & Telecommunication CSIT, Durg, (India)*

ABSTRACT

Nowadays the web pages contain a gigantic amount of information present in unstructured format and semi-structured format which can be transformed and extracted to usable information as per our requirements. The Parser provides Universal dependencies and Stanford dependencies output as well as abstract syntax tree phrase structure trees. In this paper task of Parser is explained in a very systematic manner and is applied to parse statements. Successful results were obtained.

Keywords: Parser, Stanford Parser, Part-of-Speech Tag.

I. INTRODUCTION

A Parser is a Software Component such as compiler or interpreter that breaks data into smaller elements for easy translation into another language [1]. A Parser takes input data (frequently text) in the form of a sequence of tokens or program instructions and usually builds a data structure in the form of some kind of parse tree, abstract syntax tree or other hierarchical structure. Natural language Parser can be defined as a program that works out the grammatical structure of the sentences**.** Probabilistic Parsers use knowledge of language gained from hand-parsed sentences to produce the most likely analysis of new sentences. These statistical Parsers commonly works well rather make some mistakes [1].

1.1 Stanford Parser

The Stanford Parser is a Statistical Parser or Statistical Natural Language Parser from the Natural Language Processing (NLP) group. The Stanford Parser is used to parse input data written in several languages such as English, German, Arabic and Chinese.

1.2 Overview of process

The parsing process can be done in the three phase:

I. Lexical analysis
II. Syntactic analysis
III. Semantic parsing

Steps to be applied at the time of parsing are indicated in Fig.1.

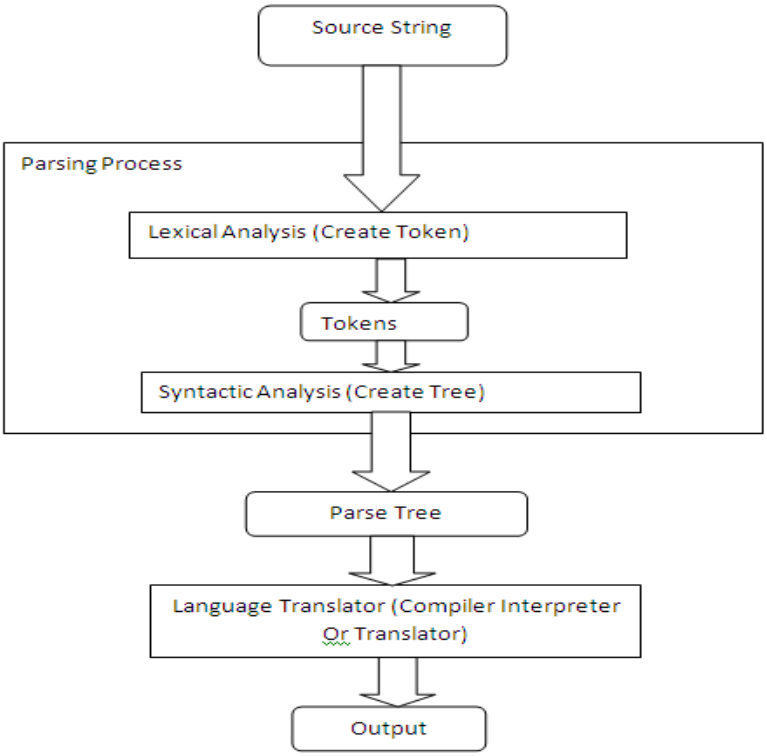

Fig. 1: Overview process of parsing

1.2.1 Lexical analysis

This is a token generation stage. Lexical analysis is a first stage of parsing process, by which the input character stream is split into meaningful symbols defined by a grammar of regular expressions (RE).

1.2.2 Syntactic analysis

The next stage of parsing process is syntactic analysis, which is checking that the tokens form an allowable expression. This is usually done with reference to a context-free grammar (CFG) which recursively defines components that can make up an expression and the order in which they must appear.

1.2.3 Semantic parsing

This phase is a final phase of parsing process. This is also known as semantic analysis, which is working out the implications of the expression just validated and taking the appropriate action.

II. TYPES OF PARSER

The task of the parser is essentially to determine how the input can be derived from the start symbol of the grammar. This can be done in two ways:

I. Top Down Parser
II. Bottom Up Parser

2.1 Top Down Parser

Top-down Parser is a parsing strategy; this can be viewed as an attempt to find left-most derivations of an input-stream by searching for parse tree using a top-down expansion where one first looks at the highest level of the

parse tree and works down the parse tree. Top-down parsing technique parses the input, and starts constructing a parse tree from the root node gradually moving down to the leaf nodes.

LL Parsers, Predictive Parser, Earley Parser and recursive-descent parser are examples of top-down Parsers.

2.2 Bottom up Parser

Bottom up Parser is a opposite of the top down Parser. It can start with the input and attempt to rewrite it to the start symbol. Bottom-up parsing starts from the leaf nodes of a tree and works in upward (Bottom to up) direction till it reaches the root node.

LR Parser, shift reduce Parser, recursive assent Parser and LR Parser are examples of bottom-up Parsers.

III. PREVIOUS WORK

Various paper related to parser are available in literature. A novel way of learning a neural network classifier for use in a greedy, transition- based dependency was indicated by Denqi Chen [3]. This Parser work very fast, while achieving 2% improvement in unlabeled and labeled attachment score on English and Chinese dataset. The Parser is able to parse more than 1000 sentences per second at 92.2% unlabeled attachment score on the English Penn Treebank. The experimental evaluations of Parser perform other greedy Parsers using features in speed and accuracy.

Syntactic parsing is a central task in natural language processing. Natural language parsing has been done with small sets of discrete categories like NP (Noun Phase) and VP (Verb Phase). Richard Soche & et all introduced a Compositional Vector Grammar (CVG), which combines parsing with Compositional Vector Grammars (PCFGs) with a syntactically untied recursive neural network that improves the PCFGs of Stanford Parser by 3.8%. The CVG evaluated in two ways: first, by a standard parsing evaluation on Penn Treebank and the by analyzing the model errors in detail. This model is linguistically more plausible since it chooses composition functions for tree. The CVG is 20% faster than the previous Stanford Parser [4].

IV. TECHNIQUES

A natural language Parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. The parse() method that invokes the current Parser object and parses the input array of strings and returns the parsed output as a string as well. The text documents are parsed and each word is assigned a Parts-Of-Speech Tags or each sentence is covered into some kind of parse tree like dependency tree, abstract syntax tree and other hierarchical structure. The parser generates the parse tree. This parse tree is given as an input to the typed dependencies generator module of the Stanford Parser. The dependency tree extracts linguistic relationships like possession, conjuction, subject, object etc. among words in a sentence[5][6].

V. RESULT

The results obtained from the Stanford Parser that takes input data (frequently text) and build a data structure- often some kind of parse tree, abstract syntax tree or other hierarchical structure.

Some Alphabetical list of part-of-speech tags are:

Table1. Part-of-speech tag [2]

S.N.	Tag	Description
1.	CC	Coordinating Conjunction
2.	CD	Cardinal Number
3.	DT	Determiner
4.	FW	Foreign word
5.	IN	Preposition and subordinating conjunction
6.	JJ	Adjective
7.	NN	Noun, singular or mass
8.	NNS	Noun, plural
9.	NNP	Proper noun, singular
10.	NNPS	Proper noun, plural
11.	PRP	Personal pronoun
12.	PRP$	Possessive pronoun
13.	RB	Adverb
14.	RBR	Adverb, comparative
15.	RBS	Adverb, superlative
16.	RP	Particle
17.	SYM	Symbol
18.	TO	*To*
19.	VB	Verb, base form
20.	VBP	Verb, non-3rd person singular present
21.	VBZ	Verb, 3rd person singular present

Query 1: The world is going digital due to the new smart technologies.

Parse Tree:

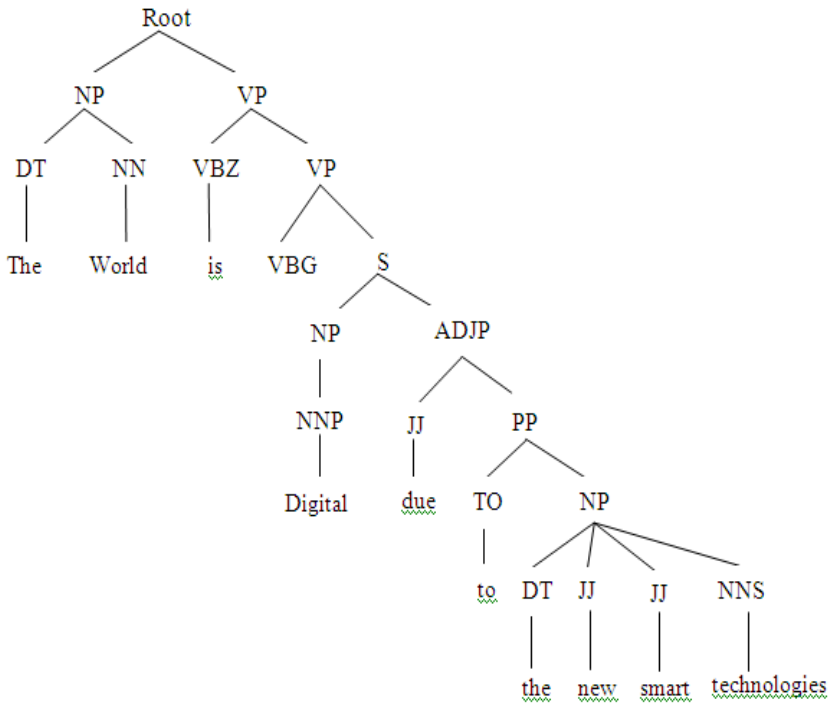

Fig. 2: Parse Tree for Query 1

Output:

(ROOT (S (NP (DT The) (NN world)) (VP (VBZ is) (VP (VBG going) (S (NP (NNP digital)) (ADJP (JJ due) (PP (TO to) (NP (DT the) (JJ new) (JJ smart) (NNS technologies))))))) (. .)))

Query 2: The Stanford Parser is a statistical natural language parser from the Stanford Natural Language Processing Group

Parse Tree:

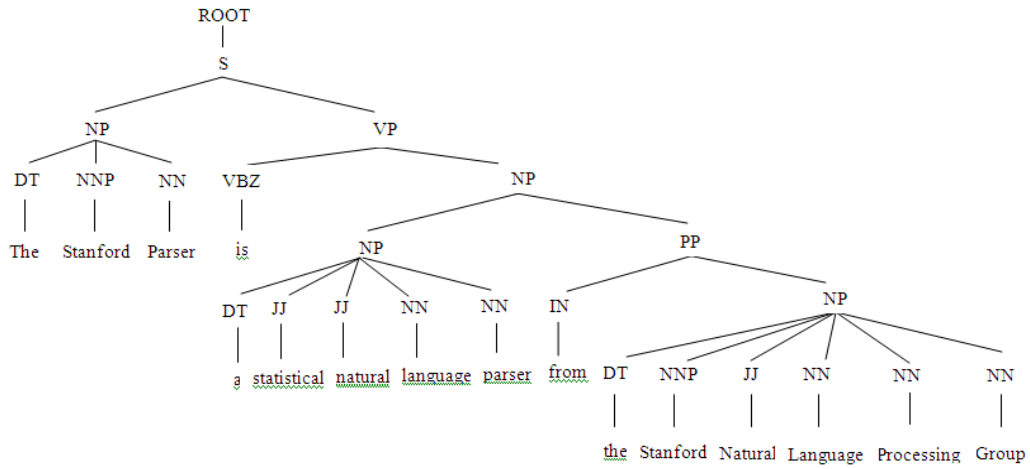

Fig. 3: Parse Tree for Query 2

Output:

(ROOT (S (NP (DT The) (NNP Stanford) (NN Parser)) (VP (VBZ is) (NP (NP (DT a) (JJ statistical) (JJ natural) (NN language) (NN parser)) (PP (IN from) (NP (DT the) (NNP Stanford) (JJ Natural) (NN Language) (NN Processing) (NN Group)))))))

VI. CONCLUSION

A Parser is software components build a data structure- often some kind of parse tree, abstract syntax tree or other hierarchical structure. The Stanford Parser is a statistical natural language Parser from the Stanford Natural Language Processing Group. This paper studies a complete overview of Parser and Stanford Parser that are used to parse or break the sentences.

REFERENCES

[1] Parsing. (2015). Retrieved from Wikipedia, the free encyclopedia website: http://en.m.wikipedia.org/wiki/parsing

[2] Penn Treebank P.O.S. Tags. (2015). Retrieved from the website: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[3] Chen, D., and Manning, C. D. 2014. A Fast and Accurate Dependency Parser using Neural Networks. Proceedings of EMNLP 2014

[4] Socher, R., Bauer, J., Christopher, D., Manning and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. Proceedings of ACL 2013

[5] Zaman H. B., Robinson P., Petrou M. (2011). Visual Informatics: Sustaining Research and Innovations. Second International Visual Informatics Conference, Selangor, Malaysia.

[6] Ghosh A ., Rajat K . De, Sankar K. Pal (2007). Pattern Recognition and Machine Intelligence. Second International Conference, PreMI 2007 Kolkata, India.