ISSN 2320 – 2602
Volume 4, No.3, March 2015
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/pdfs/2015/ijacst02422015.pdf

BASIC PARSING TECHNIQUES IN NATURAL LANGUAGE PROCESSING

Rachana Rangra[1]

[1]Bahra University, District Solan, Himachal Pradesh, India

[1]Email: rachana.rangra06@gmail.com

Madhusudan, Asst. Professor[2]

[2]Bahra University, District. Solan, Himachal Pradesh, India.

[2] Email: m9736177566@gmail.com

ABSTRACT

Parsing is the process of analyzing the sentence for its structure, content and meaning, i.e. to uncover the structure, articulate the constituents and the relation between the constituents of the input sentence. This paper briefly describes the parsing techniques in natural language processing. Parsing is the prime task in processing of natural language, as it forms the basis for all the natural language applications, like machine translation, question answering and information retrieval. We have discussed the top-down, bottom- up and the basic top-down parsing along with their issues and a brief review of the statistical and dependency parsing.

Keywords: Ambiguity, Bottom-Up, Parsing, Part-of-Speech, Natural language Processing, Top-Down.

1. INTRODUCTION

Parsing in basic terms can be described as breaking down the sentence into its constituent words in order to find out the grammatical type of each word or alternatively to decompose an input into more easily processed components. In simple terms parsing is breaking down of sentence into atomic values. To analyze data or a sentence for structure, content and meaning. For example, let's consider a sentence "John is playing game". After parsing it will be stated in terms of its constituents, as "John", "is", "playing", "game". Natural language processing applies the same concept to parse a natural language sentence. Parsing in natural language is termed as "to analyze the input sentence in terms of grammatical constituents, identifying the parts of speech, syntactic relations". Parsing is a process of determining how a string of terminals(sentence) is generated from its constituents, by breaking down of sentence into tokens. Each individual word in a sentence is termed as token. For example "John", 'is","playing", "game", are tokens for above sentence. Every natural language consist of its own grammar rules according to which the sentences are formed, parsing is used to find out the sequence of rules applied for sentence generation in that particular language. Parsing natural language sentence can be viewed as making a sequence of disambiguation decisions: determining the part-of-speech of the words, choosing between possible constituent structures and selecting labels for the constituents [4].Part-of-speech is defined as the category to which a word is assigned according to its syntactic behavior. Every language has its own part-of-speech, but here we are concerned with the part -of-speech for English Language. English language provides us with eight part of speech, viz: article, noun, pronoun, verb, adverb, adjective, preposition, conjunction. For example in the sentence, "John is playing game" part-of-speech for each token is "noun" for "John" and "game", "verb" for "is" and "playing". Making a disambiguous decision means, finding the correct part-of-speech for a word having multiple part-of-speeches, which give rise to "ambiguity". Ambiguity means having more than one interpretation of word or sentence. Example "book", it can be "noun" or "verb", depending upon its use, parsing is use to find the correct parse for a word or a sentence. Parsing results in generation of parse tree, which is the graphical representation of the order in which the grammar productions are applied during parsing of a sentence, therefore parsing can be viewed as the order in which the nodes of parse tree are constructed.

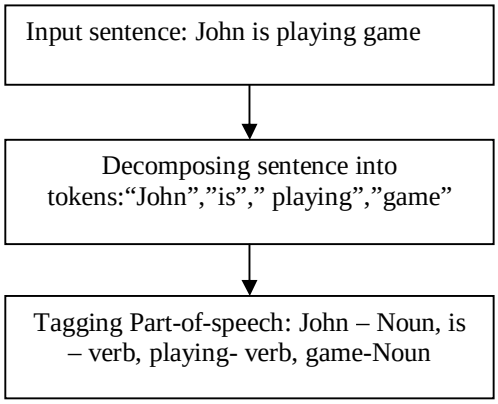


Figure1. Parsing process

2. PARSING TECHNIQUES

The basic connection between a sentence and the grammar it derives from is the parse tree, which describes how the grammar was used to produce the sentence. For the reconstruction of this connection we need a parsing technique [5].Natural Language processing provides us with two basic parsing techniques viz; Top-Down and Bottom-Up. Their name describes the direction in which parsing process advances. We have a Basic-Top-Down parsing which is the fusion of top-down and bottom-up parsing.

2.1 Top-Down parsing

Using Top-Down technique, parser searches for a parse tree by trying to build from the root node S down to the leaves. The algorithm starts by assuming the input can be derived by the designated start symbol S. The next step is to find the tops of all the trees which can start with S, by looking on the grammar rules with S on left hand side, all the possible trees are generated [7].Top down parsing is a goal directed search [7]. It tries to imitate the original production process by rederiving the sentence from the start symbol ,and the production tree is reconstructed from the top downwards [5].Top-Down parsing can be viewed as an expansion process which begins with starting symbol S, and then advances by replacing S with the Left hand side production. The common search strategy implemented in this approach is Top-Down, left-to-right and backtracking. The search starts from the root node labeled S i.e. starting symbol, construct the child nodes by applying the rules with left hand side equals to S, further expands the internal nodes using next productions with left hand side equals to internal node, if nonterminal, and continues until leaves are Part-of-speech (terminals).If the leaf nodes i.e. Part-of-speech do not matches the input string, we need to backtrack to the latest node processed and apply another production. Top-Down parsing is viewed as generation of parse tree in preorder.

Let's consider the Grammar rules

S ⟶ NP V S ⟶ NP AuxV VP

S ⟶ VP NP ⟶ Proper-Noun

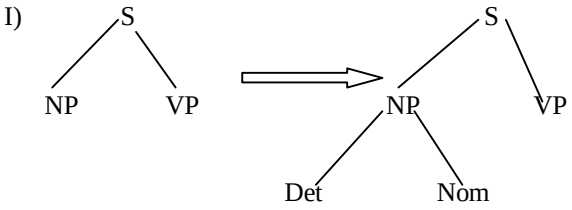
NP ⟶ Det Nominal Nominal ⟶ Noun

Nominal ⟶ Noun Nominal

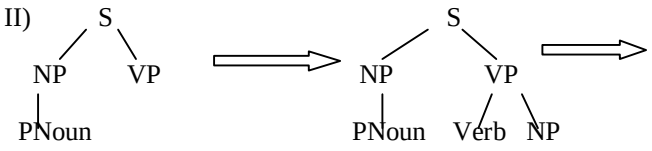
VP ⟶ Verb VP ⟶ V NP

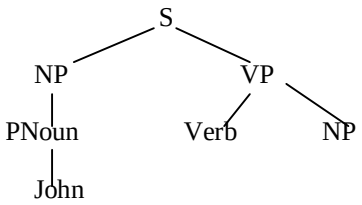
Figure.2 Grammar rules

Taking the sentence: "John is playing game", and applying Top-down parsing

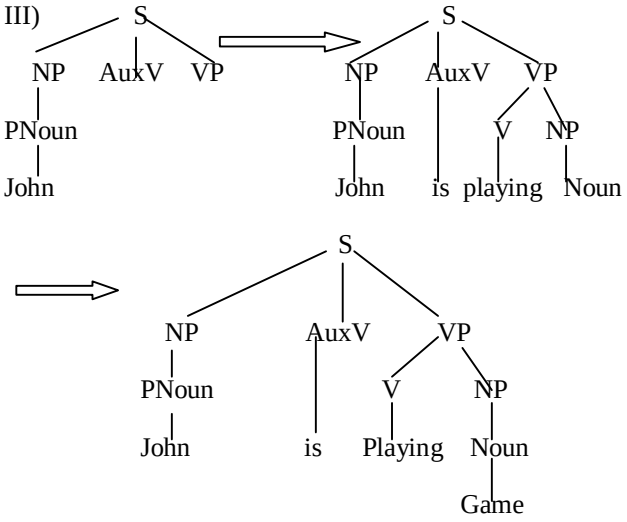


Part-of-speech does not match the input string, backtrack to the node NP





Part-of-speech Verb does not match the input string, backtrack to the node S, since PNoun is matched.



Final parse tree for the sentence

The advantage of Top-Down strategy is that it never wastes time exploring trees that cannot result in S, means it also never explores subtrees that cannot find a place in some S-rooted tree[7].Considering the other side of this approach, it has its own demerits, it leads to backtracking. The Top-Down approach spends considerable effort and time on S trees that are not consistent with the input. This weakness in Top-Down parser arises from the fact that they can generate trees before examining the input[8][6].While expanding the nonterminals it becomes difficult to decide which Right hand side production should be selected i.e. to select the appropriate starting production and further productions to avoid backtracking.

Predictive parsing is the solution for backtracking problem faced in top-Down Strategy. Predictive Parsing is characterized by its ability to use at most next (k) tokens to select which production to apply, referred to as lookahead [1].Making the right decision without backtracking .Basic idea is given A ⟶ a & A ⟶ b, the parser should be able to choose between "a" and "b". To make the correct choice it needs First(a) sets and Follow(A) sets. First(a) sets describes the set of tokens that appears as the first symbol in some string that derives from "a". Follow(A) is the set of tokens that appears immediately to the right of A in some sentential form. Predictive parsing imposes restriction on the grammar to be used i.e., the grammar must possess LL(1) property, in which the first 'L'states that we scan the input from left to right, second 'L' says we create leftmost derivation first and '1' means one input symbol of lookahead[1].Grammar should not be left recursive.LL(1) property Stated as follows:

If A ⟶ a and A ⟶ b both appears in grammar, then

First(a) ∩First(b) = Ø. This would allow the parser to make a correct choice with a lookahead of exactly one symbol.

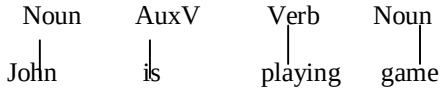
2.2 Bottom –Up Parsing

Bottom – Up parsing starts with the words of input and tries to build trees from the words up, again by applying rules from the grammar one at a time. The parse is successful if the parse succeeds in building a tree rooted in the start symbol S that covers all of the input [11].Bottom up parsing is a data directed search[7].It tries to roll back the production process and to reduce the sentence back to the start symbol S[5].It reduces the string of tokens to the starting Symbol by inverting the production, the string is recognized by constructing the rightmost derivation in reverse. The objective of reaching the starting symbol S is achieved by series of reductions, when the Right hand side of some rule matches the substring of the input string, the substring is replaced with the left hand side of the matched production, the process continues until starting symbol is reached, henceforth Bottom –up parsing can be defined as reduction process. Bottom-Up parsing can be viewed as generation of parse tree in postorder.

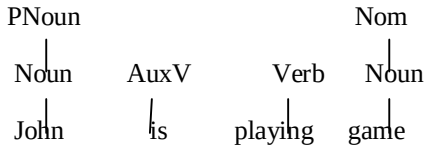
Considering the grammar rules in Figure.2 and the input sentence

"John is playing game". The Bottom-Up parsing works as follows:

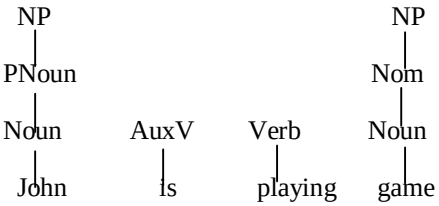
I)



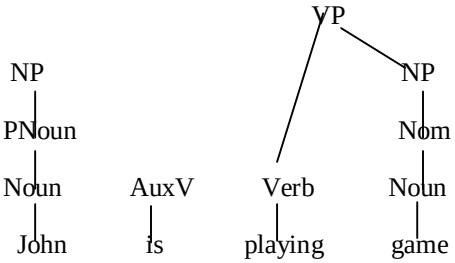
II)



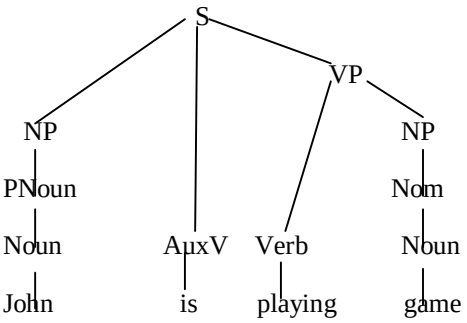
III)



IV)



V)



Bottom-Up parsing adopts the shift-reduce paradigm, where a stack holds the grammar symbol and input buffer stores the rest of the input sentence. The Shift-reduce parsing is achieved using four primary actions (1)Shift, pushes the next input symbol on the top of the stack (2)Reduce, reduces Right Hand Side of the production to its Left Hand Side

(3)Accept, successful completion of parsing and(4) Error, discover the syntax error and call the error recovery routine.

To have an operational shift-reduce parser and to determine the reducing production to be used, it implements LR parsing which uses the LR(K) grammar. Where (1) L signifies Left-to-right scanning of input (2) R indicates rightmost derivation done in reverse and (3) K, is the number of lookahead symbols used to make parsing decision. The efficiency of Bottom-Up parsing lies in the fact that, it never explores trees inconsistent with the input. Bottom-Up parsing never suggests trees that are not locally grounded in the actual input[7].However the trees have no hope of leading to an S or fitting in with any of its neighbors, are generated in abandon[1],which adds to the inefficiency of the bottom-up strategy.

2.3 Basic Top-Down Parsing

Neither of the approaches discuss above completely exploits the constraints presented by the grammar and the input words[7], therefore another technique was designed by combining the best features of Top-Down and Bottom-Up parsing, termed as Basic Top-Down parsing. The primary control strategy of Top-Down parsing is adopted to generate trees and then the constraints from the Bottom-up parsing are grafted to filter out the inconsistent parses. The parsing algorithm initiates with top-down, depth-first, left-to-right strategy, and maintain an agenda of search states, consisting of partial trees along with pointer to the next input word in the sentence. The parser takes the front state of the agenda and applies the grammar rules to the left-most unexpanded node of the tree associated with that state to produce a new set of states, and then add these new states to the front of the agenda, according to the textual order of the grammar rules that were applied to generate those states, continuing the process until either a successful parse tree is found or agenda is exhausted .Next step is to add the Bottom-up filter using left -corner rule, stated as, the parser should not consider any grammar rule if the current input cannot serve as the first word along the left edge of the derivation from that rule[7].Even though Basic Top-Down parser merges the best features of top-Down and Bottom-up strategy, yet it provides an insufficient solution to general purpose parsing problems viz: Left recursion, ambiguity and inefficient reparsing of subtrees.

3. OTHER TECHNIQUES FOR AMBIGUITY

Natural language processing provides two primary techniques for parsing, but due to their inability in resolving problem of ambiguity, different techniques were devised to resolve the ambiguity issues in parsing.

3.1 Statistical Parsing

Statistical parsing is a probabilistic parsing which resolves the structural ambiguity i.e. multiple parse trees for a sentence by choosing the parse tree with the highest probability value. The statistical parsing model defines the conditional probability, P(T|S) for each candidate parse tree T for a sentence S. The parser itself is an algorithm which searches for the tree T that maximizes P(T|S)[9].The statistical parser, uses probabilistic context-free grammars(PCFGs),context-free grammars in which every rule is assigned a probability to figure out, how to(1) find the possible parses (2) assign probabilities to them (3) pull out the most probable one[9].Statistical parsing works by using the corpus of hand -parsed text, most notably for English we have the Penn tree bank (Marcus 1993).The probability of the entire parse tree is calculated by taking the product of the probabilities for each of the rule used to construct the parse. If s is the entire sentence, π is a parse of s, c ranges over constituents of π, and r(c) is the rule used to expand c, then.

$$p(s, \pi) = \pi\, p(r(c)).$$

The probability of a rule is the product of the probability of its constituents [3].

3.2 Dependency Parsing

The fundamental notion of dependency is based on the idea that the syntactic structure of a sentence consists of binary asymmetrical relations between the words of the sentence [10] and Dependency parsing provide a syntactic representation that encodes functional relationships between words [2].The dependency relation holds between the head and the dependent. Dependency parsing uses the dependency structure representing head-dependent relations (directed arcs), functional categories (arc labels) and possibly some structural categories (parts-of-speech).

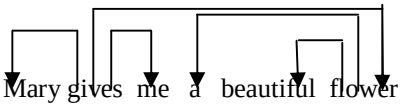


Figure.3 Dependency Tree

4. CONCLUSION

In Natural Language Processing we have two prime techniques of parsing top-down and bottom-up. Top- down strategy never explores the tree which does not result in the tree rooted in S, while bottom-up strategy do not consider the parses inconsistent with the input. But neither of them exploits the resources and provides the accurate output in case of ambiguous word, that is, they cannot handle the ambiguity issue. For dealing with the ambiguity we have Statistical Parsing based on the conditional probability and Dependency parsing works by defining the functional relationship between the words.

REFERENCES

[1]. Aho A V, Sethi R, Ullman J D. **Compilers: Principles, Techniques and Algorithm.**

[2]. Chang Ming-Wei, Quang Do, Dan Roth. **Multilingual Dependency Parsing: A Pipeline Approach.** 2006.

[3]. Charnaik Eugene. **Statistical Techniques for Natural language Parsing.** AI Magazine, 1997, 18(4).

[4]. D Magerman. **Statistical Decision-Tree Models for Parsing.** In Proc 33rd annual meeting of association for Computational Linguistics, 1995, pp.276-283.

[5]. Grune Dick, Jacobs Ceriel. **Parsing Techniques A Practical Guide.** Ellis Horwood Limited, 1990, pp. 64-66.

[6]. Irons E T. **A Syntax Directed Compiler for ALGOL 60.** Communications of the Association for Computing Machinery,1961, 4(1): 51-55.

[7]. Jurafsky D, Martin J H. **Speech and Language Processing.** Russell Stuart, Norvig Peter (eds.), Alan Apt, 2000, pp.354-372.

[8]. Kuno S. **Functional Sentence Perspective: A Case Study from Japanese and English**. Linguistic Inquiry, 1972. 269-320.

[9]. Madhusudan. **Optimizing Parsing with Multiple Pipelining**. International Journal of Engineering Research and Development, 2012, 4(3): 08-12.

[10]. Nivre Joakim, Kubler Sandra. **Dependency Parsing** Tutorial at COLING-ACL, Sydney 2006.

[11] Ynge V H. **Syntax and the Problem of Multiple Meaning**.1955.