RESEARCH ARTICLE OPEN ACCESS

Implementation Of Syntax Parser For English Language Using Grammar Rules

Shraddha Anantpure[1], Hinal Jain[2], Neha Alhat[3], Smita Bhor[4], Mrs Shanthi Guru[5]

1,2,3,4(Computer Engineering, DYPCOE, Akurdi/ University of Pune, India)
5(Assistant Prof, Computer Engineering, DYPCOE, Akurdi/University of Pune, India)

Abstract
From many years we have been using Chomsky's generative system of grammars, particularly context-free grammars (CFGs) and regular expressions (REs), to express the syntax of programming languages and protocols. Syntactic parsing mainly works with syntactic structure of a sentence. The 'syntax' refers to the grammatical and syntactical arrangement of words in a sentence and their relationship with other words. The main focus of syntactic analysis is important to find syntactic structure of a sentence which usually is represented as a tree structure. To identify the syntactic structure is useful in determining the meaning of a sentence Natural language processing processes the data through lexical analysis, Syntax analysis, Semantic analysis, and Discourse processing, Pragmatic analysis. This paper gives various parsing methods. The algorithm in this paper splits the English sentences into parts using POS (Parts Of Speech) tagger, It identifies the type of sentence (Simple, Complex, Interrogate, Facts, active, passive etc.) and then parses these sentences using grammar rules of Natural language. As natural language processing becomes an increasingly relevant, there is a need for tree banks catered to the specific needs of more individualized systems. Here, we present the open source technique to check and correct the grammar. The methodology will give appropriate grammatical suggestions.

I. INTRODUCTION

Most language syntax theory and practice is based on generative systems, such as regular expressions and context-free grammars, in which a language is defined formally by a set of rules applied recursively to generate strings of the language. Language is the important tool of communication used by the individuals. It is the tool that everyone uses to express the greater part of ideas and emotions. It shapes thought, has a structure, and carries meaning. Natural language processing is concerned with the progress of computational models of human language processing. Natural language processing, a branch of artificial intelligence that deals with the analysis and interpretation of human languages, has become increasingly relevant as people begin to rely more and more on computers for aid in communication and information compilation.

Most recent work in learning for semantic parsing has focused on "shallow" analysis such as semantic role labeling (Gildea and Jurafsky, 2002). In this paper, we address the more ambitious task of learning to map sentences to a complete formal meaning representation language (MRL). We consider two MRL's that can be directly used to perform useful, complex tasks. The first is a Prolog-based language used in a previously-developed corpus of queries to a database on U.S. geography (Zelle and Mooney, 1996). By integrating syntactic and semantic interpretation into a single statistical model and finding the globally most likely parse, an

accurate combined syntactic/semantic analysis can be obtained.

Identifying the syntactic structure is useful in determining the meaning of the sentence. The identification is done using a procedure known as parsing. Syntactic parsing deals with the syntactic structure of a sentence. In many languages, words are brought together to form larger groups termed constituents or phrases, which can be modeled using context free grammar. Context free grammar is a set of rules or productions that expresses which elements can occur in a phrase and in what order.

Chomsky's generative system of grammars, from which the ubiquitous context-free grammars (CFGs) and regular expressions (REs) arise, was originally designed as a formal tool for modeling and analyzing natural (human) languages. Due to their elegance and expressive power, computer scientists adopted generative grammars for describing machine-oriented languages as well. The ability of a CFG to express ambiguous syntax is an important and powerful tool for natural languages. Unfortunately, this power gets in the way when we use CFGs for machine-oriented languages that are intended to be precise and unambiguous. Ambiguity in CFGs is difficult to avoid even when we want to, and it makes general CFG parsing an inherently super-linear-time

problem. The detailed procedure is prescribed in various sections as follows. Section-2 provides the overview of CFG also known as phrase structure grammar as methodology of the algorithm.Section-3 represents various parsing approaches and procedures to solve this issue. Section-4 describes proposed method for checking syntax of the given sentences. Section-5 describes the experimental outcomes for the same and paper ends with conclusion and future work.

II. LITERATURE SURVEY

A Context-Free Grammar (CFG) is a mathematical system for modeling constituent structure" in natural languages, consisting of rules for the syntax of the grammar, as well as a lexicon of syntax and associated words. More formally, each rule in a CFG begins with a single start symbol, such as a type of phrase, followed by the constituents of that symbol. The constituents may be either a terminal symbol associated with a word in the lexicon (example Verb), or a non-terminal symbol, associated with a symbol defined by its own set of constituents (e.g. Noun Phrase).

Context-free grammar (CFG) was first defined for natural language by Chomsky (1957) and used for the Algol programming language by [9]. A CFG consists of four components:

1. A set of non-terminal symbols, N
2. A set of terminal symbols, T
3. A designated start symbol, S, that is one of the symbols from N.
4. A set of productions, P, of the form:A --› a

TABLE1. LIST OF ABBREVIATIONS FOR THE GRAMMAR

Abbreviations	**Abbreviations Meaning**
S	Sentence
Det	Determiner
Adj	Adjective
Pron	Pronoun
Num	Numerals
Conj	Conjunction
Neg	Negation
Prep	Preposition
Adv	Adverb
V	Verb
N	Noun
NP	Noun Phrase
VP	Verb Phrase
NPP	Noun Preposition Phrase
VPP	Verb Preposition Phrase

III. PARSING APPROACHES

A Context Free Grammar (CFG) defines the syntax of a language but CFG does not specify how structures are assigned. Parsing is the task that uses the rewrite rules of a grammar to either generate a particular sequence of words or reconstruct its derivation or phrase structure tree. Phrase is a phrase structure tree which is constructed from a sentence. There are three parsing approaches: 1) Top-Down Parsing and 2) Bottom-Up Parsing 3) Shift-Reduce Parsing.

A. Top-down parsing

Top down parsing starts the searching from the root node say S and works downwards towards the leaves that is the input can be derived from the chosen start symbol S, of the grammar. The next step is to find all sub-trees which can start with start symbol S. To generate the sub trees of the second -level search, we expand and root node using all the grammar rules associated with S on their left hand side. In a same way, each non-terminal symbol in the resulting sub-trees is expanded next using the grammar rules having a matching non-terminal symbol on their left hand side. The right hand side of grammar rules provides the node that is to be

generated, which are then expanded recursively. As the expansion continues, the tree grows downward and eventually reaches a state where the bottom of the tree consists only of part-of speech categories. At that point, all trees whose leaves do not match with the words in the input sentence are rejected, leaving only trees that represent successful parses.

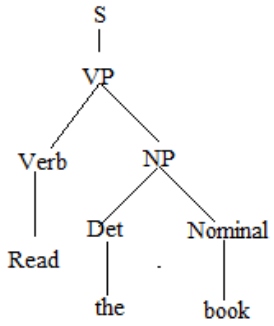

Figure1. Top-Down Search Space

B. Bottom-up parsing

A bottom-up parser starts with the words in the input sentence and attempts the construction of parse tree in an upward direction towards the root node say S. In each step, the parser looks for the rules in the grammar where the right hand side matches some of the production in the parse tree constructed so far, and reduces it using the left hand side of the production. If the parser reduces the tree to the start symbol S of the grammar then the parse is considered successful. Each of these parsing approaches has its own advantages and disadvantages. The top-down search starts generating the trees with the start symbol S. The grammar never wastes time exploring a tree leading to a different root. However, it wastes the considerable time in exploring S trees that eventually result in words that are inconsistent with the input. This is because a top down parser generates trees before seeing the input. On the other hand, a bottom-up parser never explores the tree that does not match with the input but it wastes the time in generating trees that have no chance of leading to an S-rooted tree. There are many attempts have been made to develop a syntax parsing with various approaches. Majority of approaches to check syntax correctness is based on probabilistic approach.

C. Shift-Reduce Parsing

In order to build a parser, we need to create an algorithm that can perform the steps in the above rightmost derivation for any grammar and for any input string. Every CFG turns out to have an automaton that is equivalent to it, called a pushdown automaton (just like regular expressions can be converted to finite state automata). A pushdown automaton is simply a finite-state automaton with some additional memory in the form of a stack (or pushdown). This is a limited amount of memory since only the top of the stack is used by the machine. This provides an algorithm for parsing that is general for any given CFG and input string. The algorithm is called shift-reduce parsing which uses two data-structures: a buffer for input symbols and a stack for storing CFG symbols and is defined as follows:

1. Start with an empty stack and the buffer contains the input string.
2. Exit with success if the top of the stack contains the start symbol of the grammar and if the buffer is empty.
3. Choose between the following two steps (if the choice is ambiguous, choose one based on an oracle):

Shift a symbol from the buffer onto the stack. If the top k symbols of the stack are _1 : : : _k which corresponds to the right-hand side of a CFG rule A ! _1 : : : _k then replace the top k symbols with the left-hand side non-terminal A.

4. Exit with failure if no action can be taken in previous step.
5. Else, go to Step 2.

IV. ALGORITHM

4.1 Rule Based Algorithm

The WORKING of syntax analyzer is done through following rule based algorithm.

1. Enter a sentence.
2. Categorize the sentence using Table-2.

3. Check the phrases of sentences using various tags that are returned by POS tagger. (Its noun phrases (N, NP, NPP) and verb phrases (V, VP, VPP)).
4. Partition the sentence into NP and VP identified in Table-4.
5. Parse the NP, NPP, V and VPP by matching it against the Grammar rules.
6. If all parts of the sentences are parsed correctly then sentence is syntactically correct, else the sentence is syntactically incorrect.

TABLE2. CATEGORIZATION OF ENGLISH SENTENCES

Basis of categorization	**Category**
Sentence with only one subject, one verb and one object.	Simple
Sentence with only one subject, verb, and adjective followed by a verb.	SVO with adjective
Sentences with more than one subject or object and having "and"…"or" in it.	Complex
Sentence terminating with a"?"	Interrogative
Sentences containing conjunctions.	Conjunctions
Sentences starting with This, That.	Facts
Simple Sentences.	Active
Sentences in which the subject follows "by".	Passive

4.2 POS TAGGER

A Part-of-Speech Tagger (POS Tagger) is a part of software that reads the text in some language and allocates the parts of speech (i.e. tags) to each word. It assigns a part-of-speech like noun, verb, pronoun, preposition, adverb, and adjective or other lexical class marker to each word in a sentence. This software is a Java implementation of the log-linear part-of-speech taggers. There are number of Taggers like Stanford Tagger, Apache UIMA Tagger; Eric Brill's simple Rule Based Tagger etc. Out of which Stanford tagger has been used. Its basic download contains two trained tagger models for the English. The full download contains three trained English tagger models that are an Arabic tagger model, a Chinese tagger model, and a German tagger model. Both the versions include the same source and the other required files. The tagger can be retrained on any language, given POS-annotated training text for the language. The input to a tagging algorithm is a string of words of a natural language sentence and a quantified tag set (a finite list of Part-of-speech tags). The output is a single finest parts-of-speech tag for each term as shown in table-3.

TABLE3. POS TAGGED OUTPUT AND THEIR MEANINGS.

Tagger output	**Meaning**	**Tagger output**	**Meaning**	**Tagger output**	**Meaning**
CD	Cardinal Numb	NNPS	Proper Noun	TO	to

	er		, Plural		
CC	Coordinating Conjunction e.g. and, but, or..etc.	NNS	Noun, plural	VBN	Past participle
DT	Determiner	PDT	Predeterminer e.g. all, both.. when they precede an article	UH	Interjection e.g. uh, well, yes, my..
EX	Existential There	POS	Possessive Ending e.g. Nouns ending in ‘s	VB	Verb, base form subsumes imperatives, infinitives and subjunctives
FW	Foreign Word	PRP	Personal Pronoun e.g. I, me, you, he..	VBD	Verb, past tense includes the conditional form of the verb to be
IN	Preposition or subordinating conjunction	PRP $	Possessive Pronoun e.g. my, your, mine, yours	VBG	Verb, gerund or present participle
JJ	Adjective	RB	Adverb	VBP	Verb, non-3rd

			Most words that end in -ly as well as degree words like quite, too and very		person singular present
JJR	Adjective, comparative	RBR	Adverb, comparative, adverbs	VBZ	Verb, 3rd person singular present
JJS	Adjective, superlative	RBS	Adverb, Superlative	WDT	Wh-determiner e.g. which, what, who
MD	Modal e.g. can, could, might, may	SYM	Symbol used for mathematical, scientific symbols	WPS	Possessive Wh-pronoun
NN	Noun singular or mass	WRB	Wh - adverb e.g. how, where, why	NNP	Proper Noun, singular
LS	List Item Marker	RP	Particle	WPS	Possessive wh-pronoun

4.3 Categorization Based On Kind Sentence And Grammar Rules

According to Wren and Martin the sentence is comprises of Subject, Verb and Object. So that, each sentence has a subject(S), Object (O) and a Verb (V). Some sentences may also have adjectives, adverbs and conjunctions. There are also sentences which are interrogative i.e. the sentences that ask a question. Keeping all these in mind, sentences are categorized into different type. It is important to categorize sentences as the POS

tagger treats the sentences as group of words. It does not look at the meaning of the sentence as a whole. The basic process of categorization is shown in table2. The categorization is as follows:

1. Sentences having exactly one subject, one verb and one object. (Simple)
2. Sentences having exactly one subject, one verb, one object and adjectives also. (Simple with ADJECTIVES).
3. Sentences containing more than one noun and verbs. (COMPLEX)
4. Sentences contains question. (INTERROGATIVE)
5. Sentences containing conjunctions. (CONJUCTIONS)
6. Simple fact statements. (FACTS)
7. Sentences in active form. (ACTIVE)
8. Sentences in passive form. (PASSIVE)

The categorization has been made to check for the accuracy of the system with respect to the types of sentences. After categorizing the sentences the format of sentences using POS tagger is checked. POS tagger identifies the noun phrases (N, NP, NPP) and verb phrases (V, VP, VPP) using the tags mentioned in the Table-3. Then partition the sentences into different phrases like NP and VP defined in Table-4. Then it parses the NP, NPP, V and VPP by matching it with the Grammar rules. Grammar rules (from Table-4) have been implemented for English language sentences and are identified that they are working for different types of sentences like Simple, complex, active, passive etc. using table-2. The grammar rules to be checked for the syntax analyzer are as shown in table4.

TABLE4. CATEGORIZATION OF ENGLISH SENTENCES

Sr. No.	**Phrases**	**Phrases and Rules**
1.	S	i. S = NP VP ii. S = NPP VP iii. S = VP iv. S = NP NPP VP v. S = NPP NPP NP VP
2.	NP	i.NP = N ii. NP = Det Adj N iii. NP = Det N iv. NP = Pron v. NP = Pron N vi. NP = Num N vii. NP = Num N N viii. NP = N Conj N ix. NP = Num N N Conj N x. NP = Det N N xi. NP = Det Adj Adj N xii. NP = Pron N N xiii. NP = Adj Pron N xiv. NP = Det Adj N N xv. NP = Det Adj N Pron xvi. NP = Neg N xvii. NP = Pron Adj N
3.	NPP	NPP = Prep NP
4.	AP	i. AP = Adj ii. AP = Adj Adj iii. AP = Adj Conj Adj
5.	APP	APP = Prep AP
6.	V	i.V = V ii. V = V V

		iii. V = V Adv V iv. V = V Neg V v. V = V V V V vi. V = V Conj V vii. V = V Adv viii. V = V Neg V Adv ix. V = Adv Conj Adv x. V = Adv V Neg V xi. V = V Adv Conj Adv xii. V = Adv V xiii. V = V V Adv
7.	VPP	VPP = Prep V
8.	VP	i.VP = V NP ii. VP = V VPP NP iii. VP = V NPP NP iv. VP = V NP NPP v. VP = V AP vi. VP = V NP NP VPP vii. VP = V viii. VP = V NPP ix. VP = V VPP x. VP = V NP V xi. VP = V NP VPP NP xii. VP = V VPP NPP xiii. VP = V NP NPP V NP xiv. VP = V NP AP xv. VP = V NP AP VPP xvi. VP = V NPP NPP xvii. VP = V NP V NPP xviii. VP = V VPP NP NP xix. VP = V NP NPP NPP xx. VP = V NPP NPP NPP xxi. VP = V VPP AP NPP NPP xxii. VP = V VPP NP NPP xxiii. VP = V AP NPP NPP xxiv. VP = V NP AP NPP xxv. VP = V NPP AP xxvi. VP = V VPP NP AP xxvii. VP = V AP NPP xxviii. VP = V NP VPP NP NPP xxix. VP = V NP NPP xxx. VP = V NPP VPP NP xxxi. VP = V NPP AP NPP

The analysis of words into the sentence is to know the grammatical structure of the sentence. The words are converted into the constructions that show how the words relate to each other. Some of the sentences may be prohibited if they disrupt the rules of the language for how words may be combined.

V. **RESULTS**

Experimentation of different samples is chosen such as a word, a sentence, a paragraph and is also chosen for different categories of sentences such as simple, complex, active, passive voice, questions etc. The algorithm of Stanford POS tagger does the tokenization of input query. After the stop word removal, spelling checking of each word is been done. Followed by sentence categorization based on pre-defined rules. We studied the suggestions for a given sentence and correction. We have also studied the keyword extraction by using page

rank algorithm based on occurrences and priority of the keywords. The sample sentences and their corresponding syntactic understanding whether they are syntactically correct or not shown in the table5.

TABLE5. RESULTS OF SYNTAX ANALYSER

Type of sentence	**Sample Sentences**	**Output**
Simple	1. The angry girl kicked the ball. 2. She went to school. 3. I want to know your name. 4. They lived in a huge palace.	Sentence is syntactically correct.
Simple + ADJ	1. Rahul is a clever boy. 2. He likes tasty pizza. 3. I love fresh flowers. 4. Jack likes to visit lovely places.	Sentence is syntactically correct.
Complex	1. They were having a good time. 2. They were playing in the ground 3. They were studying in the good college. 4. He was selling fruits in front of the hall.	Sentence is syntactically correct.
Questions	1. Who are you? 2. What is your name? 3. When is your birthday? 4. What is the name of your village?	Sentence is syntactically correct.
Conjunctions	1. He was put behind the bars for his crime. 2. The cat was sitting under the chair. 3. The children performed fabulously in the concert. 4. They went to the park and played football.	Sentence is syntactically correct.

Facts	1. Hellen Keller was blind. 2. Sun rises in the east. 3. The earth is round.	Sentence is syntactically correct.
Active sentences	1. The girl was washing the car. 2. Sita writes a letter. 3. Rita wrote a letter. 4. Rahul has written a letter.	Sentence is syntactically correct.
Passive sentences	1. The car was being washed by the girl. 2. A letter is written by Sita. 3. A letter has been written by Teena.	Sentence is syntactically correct.
Incorrect sentences	1. Sita a letter. 2. Rita wrote a. 3. The girl washing the car. 4. Boy the go the to store	Sentence is syntactically incorrect.

VI. CONCLUSION AND FUTURE WORK

We studied the syntax analysis, syntax Representation for English Language, POS tagging technique, Sentence categorization and an approach to check syntactic correctness of the sentence. On the basis of suggestions popped selection of the correct suggestion. We also studied keyword searching and extraction of its meaning using page rank algorithm. By increasing the domain into universal the accuracy can be increased gradually.

REFERENCES

[1] *Syntax Parsing: Implementation using Grammar-Rules for English Language - 2014 International Conference on Electronic Systems*, Signal Processing and Computing Technologies.

[2] STANFORD POS TAGGER: nlp.stanford.edu/software/tagger.shtml I.S. Jacobs and C.P. Bean, "*Fine particles, thin films and exchange anisotropy*," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[3] Bharti Akshar and Rajeev Sangal, "*A Karaka-based approach to parsing of Indian languages*", Proceedings of the 13th Conference on Computational Linguistics, Association for Computational Linguistics.R. Nicole, "*Title of paper with only first word capitalized*," J. Name Stand. Abbrev., in press.

[4] Wren and martin, English grammar and composition. S. Chand & Company LTD.

[5] Jurafsky, Daniel and James H. Martin, "*Speech and Language Processing:An Introduction to Natural Language Processing*" Computational Linguistics, and Speech Recognition, Prentice Hall, NJ, , 2000.

[6] Claire M. Nelson, Rebecca E. Punch, John Donaldson, "*An Interactive Software Tool for Parsing English Sentences*", Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics, 2011.

[7] Bharti Akshar, Vineet Chaitanya, and Rajeev Sangal, *Natural Language Processing: A Paninian Perspectiue*, Prentice-Hall of India, 1995.

[8] Charniak, Eugene, *Statistical Language Learning*, MIT press, Cambridge, 1993.

[9] Charniak, Naur, Peter,J.w. Backus , F.L. Bauer,J. Green , C.Katz,J. McCarthy, A.Perlis, H. Rutishauser, K. Samelson, B. vauquois, J. H. wegstein, A.van Wijngaarden, and M. Woodger, 'Report on the algorithmic language ALGOL 60,' communications of the ACM, 3(5) pp. 299-314,1960.