A Structural Algorithm for Complex Natural Languages Parse Generation

Enikuomehin, A. O.
Dept. of Computer Science,
Lagos State University,
Lagos, Nigeria

Rahman, M. A.
Dept. of Computer Science,
Lagos State University,
Lagos, Nigeria

Ameen A. O.
Dept. of Computer Science,
University of Ilorin,
Ilorin, Nigeria

***Abstract*— In artificial intelligence, the study of how humans understand natural languages is cognitive based and such science is essential in the development of a modern day embedded robotic systems. Such systems should have the capability to process natural languages and generate meaningful output. As against machines, humans have the ability to understand a natural language sentence due to the in-born facility inherent in them and such is used to process it. Robotics requires appropriate PARSE systems to be developed in order to handle language based operations. In this paper, we present a new method of generating parse structures on complex natural language using algorithmic processes. The paper explores the process of generating meaning via parse structure and improves on the existing results using well established parsing scheme. The resulting algorithm was implemented in Java and a natural language interface for parse generation is presented. The result further shows that tokenizing sentences into their respective units affects the parse structure in the first instance and semantic representation in the larger scale. Efforts were made to limit the rules used in the generation of the grammar since natural language rules are almost infinite depending on the language set.**

Keywords—Natural Language; Syntax; Parsing; Meaning Representation

I. INTRODUCTION

Natural languages [1,2] are used in our everyday communication. They are commonly referred to as human languages. Humans are able to process natural languages easily because it is their basic language of communication since birth. The human system has the capability to learn and use such languages and improve on it over time. Recently, there has been renewed effort in developing systems that emulate human due to increased service rendering requirements including several efforts in [3].

A major factor to be considered in such system is that, they must have the capability to act like human. The need includes the ability to process human speech,(Speech Recognition, an area that has had great research attention) in a way that it can receive speech signals, converts it into text, processes the text and provides a response to the user. The user is obviously more comfortable using his or her natural language to present such speech. However, natural language is a very complex language due to the high level of ambiguity existing in it. This is one of many factors, others include the availability of large set of words in several unstructured order. Thus, to make a functional system, these issues must be clearly addressed. Processing natural languages involves the concept of interpretation and generalization [4]. In Interpretation, the process involves understanding the natural languages while generalization is a next to interpretation handles the representation of the interpreted language. The process of representation will only be functional if the language of presentation is understood by the system. In understanding such languages, several stages of operations are involved. They include morphological analysis (how words are built from morphemes, a morpheme is the smallest meaningful unit in the grammar of a language), chunking (breaking down sentences into words known as tokens, a token is a symbol regarded as an individual concrete mark, not as a class of identical symbols, *it* is a popular alternative to full parsing), syntactic analysis (analyzing the sentences to determine if they are syntactically correct) and semantic analysis (looking into the meanings). One can consider the importance related to the representation in morphemes as stated above, using the following example, Consider the word "Unladylike" This word consists of three morphemes and four syllables. The Morpheme breaks into: un- 'not', lady '(well behaved) female adult human', like 'having the characteristics of'. None of these morphemes can be broken up any more without losing all the meaning the word is trying to convene. *Lady* cannot be broken up into "la" and "dy," even though "la" and "dy" are separate syllables. Note that each syllable has no meaning on its own.

Thus, our representational framework can be determined by the morphology existing in any given word. This process can be manually interpreted but as the set of terms to be considered increases, the manual interpretation has greater tendencies to fail. Thus an appropriate scheme is to introduce algorithms that can handle such complex representation of natural language in a way that appropriate parse needed for machine translation of natural language can be generated. Such algorithm will generates syntactic structures for natural language sentences by producing a syntactic analysis of any given sentence correctly whose output is the syntactic structure represented by a syntax tree. The syntax tree shows how words build up to form correct sentences. Children learn language by discovering patterns and templates. We learn how to express plural or singular and how to match those forms in verbs and nouns. We learn how to put together a sentence, a question, or a command. Natural Language Processing assumes that if we can define those patterns and describe them to a computer then we can teach a machine something of how we speak and understand each other. Much of this work is

based on research in linguistics and cognitive science. A sentence then has to be parsed for syntactic analysis. Thus, the need for the appropriate algorithm that can handle the parsing of complex natural language sentences.

In this paper, the discussions above were considered and we present an algorithm using the UML (unified modelling language) to parse natural language sentences. This model depicts various aspects of the algorithm which includes:

- An association diagram that shows the major components in our system and how they associate with one another.
- A dependency diagram that shows how each component depends on the other in order to be able to carry out its own work.
- A class diagram that depicts each component in terms of classes showing its members and methods.
- A pseudo code to show the major steps involved in each component.

A scalable interface showing the implementation of the algorithm was developed and tested to determine the level of correctness of the output.

II. Background And Earlier Work

Natural Language Processing (NLP) is the capacity of a computer to "understand" natural language text at a level that allows meaningful interaction between the computer and a person working in a particular application domain [5]. The natural language processing concepts involves the use of many tools which are essentials of developing a man-like machine. This tools includes some programming languages such as Prolog, Ale, Lisp/Scheme, and C/C++. The tools are formulated appropriately within some defined concepts using Statistical Methods - Markov models, probabilistic grammars, text-based analysis and also Abstract Models such as Context-free grammars (BNF), Attribute grammars, Predicate calculus and other semantic models, Knowledge-based and ontological methods [6].

In this paper, we focus on the generation of appropriate parse structure for any natural language sentence. This step is considered as a major step in the natural language research domain. Syntactic analysis majorly involve the structure of a given natural language sentence presented by retrieving it in a structural manner with the rules of forming the sentences, and the words that make up those sentences. This is also includes the grammar and lexicon. It involves morphology that is the formation of words from stems, prefixes, and suffixes. Syntactic analysis shows the legal combination of words in a sentence. Generating syntactic structure involves the use of grammar, that is, the rules for forming correct sentences. Natural languages have to be parsed to obtain the syntactic information encoded in them. But natural language is ambiguous which necessitated the intervention of the use of an algorithm. This structure will present the analysis of a sentence by showing how words combine correctly to form valid phrases and how this phrase legally build up sentences. A parsing algorithm operates based on some set of rules known as grammar that tells the parser valid phrases and words in a sentence. The ambiguity of natural languages leads to a complex analysis of it, so it is more suitable to use a parsing algorithm in situations where the natural language sentence is complex. In such cases, a sentence generates multiple parse trees for the same natural language. As natural language understanding improves, computers will be able to learn from the information online and apply what they learned in the real world. Combined with natural language generation, computers will become more and more capable of receiving and giving instructions. Ambiguities are a problem because they can lead to two or more different interpretations of the same word. They are often part of the subconscious knowledge, so requirements writers will not necessarily recognize these potential sources of misunderstandings. There are different kinds of ambiguity. *Lexical ambiguity* refers to single expressions that may be reasonably interpreted in more than one way.

The study of natural language processing also incorporates other fields such as linguistics and statistics. The knowledge of linguistics provides the grammars and vocabularies needed and the knowledge of statistics provide mathematical models that the algorithm for processing natural languages uses. Various algorithms had been developed in time past for natural language processing and more algorithms are currently under development to solve more of natural language processing problems. In 1950, Alan Turing [7] proposed "Turing Test" in his famous article "Computing Machinery Intelligence". Turing test is a test that is used to know the ability of computer systems to impersonate humans. In 1954, the George Town experiment came up which involved a full automatic translation of more than sixty Russian sentences into English. In addition, in 1960s, some successful natural language processing systems were developed. These systems majorly include: ELIZA, [8,9]. SHRDLU [10]., a system that works in restricted blocks with restricted vocabularies that can be used to control robotic arms. Many programmers began to write conceptual ontologies in 1970, they are structured to appropriate real-world information into computer system. Up to the 1980s, most natural language processing systems were based on complex sets of hand-written rules.

Furthermore, in 1980s [4], the first "statistical machine translation systems" was developed. At this time, there was a great revolution in natural language processing with the introduction of "machine learning algorithms" for language processing. This is as a result of the increase in computational power resulting from the application of Moore's law [11]. Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. Statistical methods are used in NLP for a number of purposes, e.g., for word sense disambiguation, for generating grammars and parsing. At the core of any NLP task there is the important issue of natural language understanding. The process of building computer programs that understand natural language involves three major problems: the first one relates to the thought process,

the second one to the representation and meaning of the linguistic input, and the third one to the world knowledge. Thus, an NLP system may begin at the word level – to determine the morphological structure, nature (such as part-of-speech, meaning etc.) of the word – and then may move on to the sentence level – to determine the word order, grammar, meaning of the entire sentence, etc.— and then to the context and the overall environment or domain. A given word or a sentence may have a specific meaning or connotation in a given context or domain, and may be related to many other words and/or sentences in the given context. Automatic text processing systems generally take some form of text input and transform it into an output of some different form. The central task for natural language text processing systems is the translation of potentially ambiguous natural language queries and texts into unambiguous internal representations in which matching and retrieval can take place. Masaru Tornita (1984) [3]," proposed that "When a parser encounters an ambiguous input sentence, it can deal with that sentence in one of two ways. One way is to produce a single parse which is the most preferable. Such parsers are called one-path parsers. On the other hand, parsers that produce all possible parses of the ambiguous sentence are called all-paths parsers". A suitable parser for parsing natural languages is one that generates several parses or parses trees for a natural language sentence because a sentence can have a syntax and different meaning. NLP systems, in their fullest implementation, make elegant use of this kind of structural information. They may store a representation of either of these sentences, which retains the fact that Chelsea won Benfica or vice versa. They may also store, not only the fact that a word is a verb, but the kind of verb it is.

One-path parsers are, naturally, much faster than all-paths parsers because they look for only one parse. There are, however, situations where all-paths parsers should be used. MLR is an extension of LR. The LR parsing algorithm, however, has seldom been used for natural language processing, because the LR parsing algorithm is applicable only to a small subset of context-free grammars, and usually it cannot apply to natural languages. Though the efficiency of a LR parsing algorithm is preserved, MLR parsing algorithm can apply to arbitrary context-free grammars, and is therefore applicable to natural languages. We cannot directly adopt the LR parsing technique for natural languages because not all context-free phrase structure grammars (CFPSG's) can have an LR parsing table. Only a small subset of CFPSG's called *LR* grammar can have such an LR parsing table. Every ambiguous grammar is not LR, and since natural language grammars are almost always ambiguous, they are not LR therefore we cannot have an LR parsing table for natural language grammars. Liddy (1998) and Feldman (1999) [5] suggest that in order to understand natural languages, it is important to be able to distinguish among the following seven interdependent levels, that people use to extract meaning from text or spoken languages:

- Phonetic or phonological level that deals with pronunciation
- Morphological level that deals with the smallest parts of words, that carry a meaning, and suffixes and prefixes
- Lexical level that deals with lexical meaning of words and parts of speech analyses
- Syntactic level that deals with grammar and structure of sentences
- Semantic level that deals with the meaning of words and sentences
- Discourse level that deals with the structure of different kinds of text using document structures and
- Pragmatic level that deals with the knowledge that comes from the outside world, i.e., from outside the contents of the document.

The above justification seems sufficient for the development of an appropriate model for implementing an algorithm for parse generation of natural language sentences.

III. MODEL

We present a model to show the major components in our algorithm and how they interact in order to generate effective parse results for complex natural language sentences. To parse a natural language sentence (syntactic analysis), the most important things to consider are:

- The parser (the algorithm)
- Set of grammars for the language (the rules of correct syntax)

Based on our model, the major components used are: Tokenizer - which breaks down a given sentence into words usually known as tokens, Part of speech tagger - represented as those whose function is to tag each word to their respective part of speech. Parse- which analyses the sentence to check if it conforms to some sets of grammar (English grammar) for the language of the input sentence and finally the ParseTree, The parse tree represents the graphical nature of the natural language. The UML Association diagram is necessary to visualize the association between the classes. The UML Class diagram is used to visually describe the problem domain in terms of types of object (classes) related to each other in different ways. There are three primary inter-object relationships: *association*, *aggregation*, and *composition*. Using the right relationship line is important for placing implicit restrictions on the visibility and propagation of changes to the related classes.

Following from above, the formulated PSEUDOCODE is then presented as:

```
Class Tokenizer
//variable declaration
String sentence
Int i,tokenLength // i is a counter

Sentence=get sentence from user
Break sentence to tokens//break sentence to words
tokenLength=get number of words in sentence
```

```
//create two arrays to store the words and their part of speech
New tokenArray(tokenLength)
New posArray(tokenLength)
tokenArray=tokens//store tokens in array

//tag words to their part of speech
For(i=0;  i<tokenLength; i++)
{          posArray[i]=posTagger.tagWord(tokenArray[i])
}

//parse sentence and get parse tree
parseTree=parser.parseSentence(tokenArray,posArray)

display parseTree


Class POSTagger
//Variable declaration
String pos

//create an array of words and their part of speech
New dictionary(l)//where l is number of words in dictionary
New partOfSpeech(l)

Function tagWord(String word)
{
      For(i=0; i<l;  i++)
      {
          If(dictionary[i]=word)
          {
             Pos=partOfSpeech[i]
              Return pos
           }
      }

Class parser

//variable declaration
int number of words

function  parseSentence(String[] words, String[] pos)
{
  numberOfWords=words.getNumberOfWord

  if(numberOfWords=3)
  {
     If(words follow grammer 1)
         Draw parse tree 1
    Else if(words follow grammer 2)
         Draw parse tree 2
       |
       |
    Else if(words follow grammer n)
        Draw parse tree n
  }
 Else if(numberOfWords=4)
  {
     If(words follow grammer 1)
         Draw parse tree 1
    Else if(words follow grammer 2)
         Draw parse tree 2
       |
       |
    Else if(words follow grammer n)
        Draw parse tree n
  }
  |
  |
 Else if(numberOfWords=n)
 {
     If(words follow grammer n)
         Draw parse tree n
 }
}
```

IV. Implementation And Result

The model is implemented as a stand-alone application using the java programming language. The application was designed such that only an input sentence of a maximally defined number of terms can be accommodated. When a user enters a sentence within the specified limit, the system verifies the correctness of the sentence and then outputs a graphical display of the parse tree for the sentence. The resulting output is shown in figure 1.

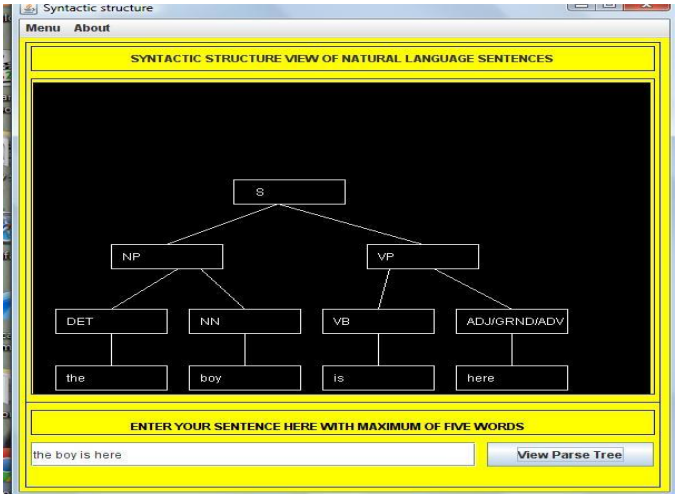

Fig.1. Simple java based parser

Figure 1.6 shows the parsing of a natural language sentence based on codes developed in java. The generation is extended by implementing an internal scheme based on a dictionary such that when a word is not present in the application's dictionary, the structure of the surrounding words can be used to tell the possible part of speech the word will belong to. For example, the word "here" shown in figure 1.6 is not in the application's dictionary yet it was tagged as either adjective or gerund or adverb, this is because only these categories of words can occupy that position once the preceding words follows the order "determinant noun verb".

Figure 2 shows another generation of the parser interface where the rule implements the sentence format "noun verb gerund".

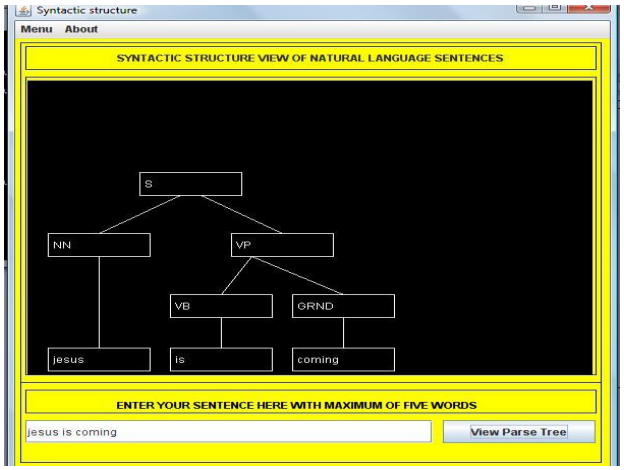

Fig.2. Noun Gerund parse

If a user enters an invalid sentence or a sentence whose grammar is not present in our list of grammars, the system will output the following:

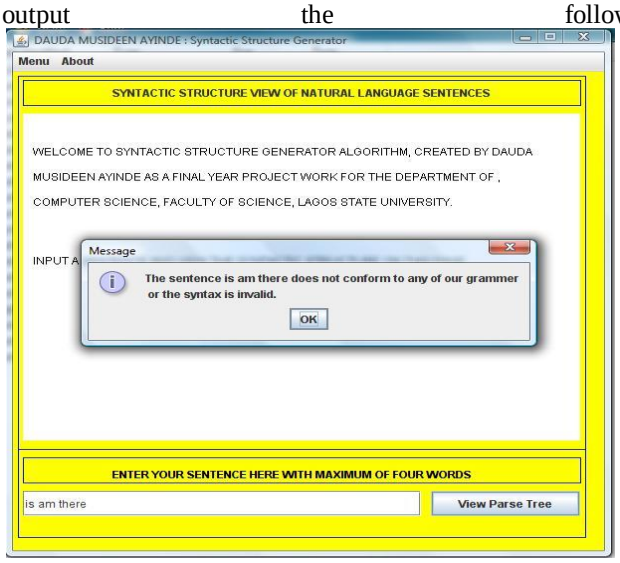

Fig.3. Parsing non defined sentence grammar

V. CONCLUSION

The algorithm presented in this paper can be extended based on the required complexity of the system. The defined process for tokenization and the use of a natural language interface in solving parse generation has shown the effectiveness of a well posted algorithm in solving the natural language parse generation problem. An extension of this work is in its ability to generate one parses tree even when the observed ambiguity is high. Parsing natural language is a complex task, an efficient algorithm for parsing natural language has been shown in this work as a necessary tool even within the inherent complexity observed.

An extended form of the LR parsing algorithm though not discussed in this paper will also be an efficient algorithm as it will generate multiple parses for natural language sentences. Such is similar to multiple application of the algorithm presented in this paper and can be called the MLR parsing algorithm.

REFERENCES

[1] J. S. Amberg, Introduction: what is language", The American journal of English, history, structure and usage, page 1-10. (1987
[2] J. Lyons, Natural Ianguage and Universal Grammar. New York: Cambridge University Press. pp. 68–70. ISBN 978-0521246965. (1991).
[3] P.Pantel, T. Lin, and M.l Gamon, Mining Entity Types from Query Logs via User Intent Modeling, Association for Computational Linguistics, July 2012
[4] G,G. Chowdhury, Natural language processing", The journal of department of computer and information science, university of strathclyde, page1-22. ,(1991)"
[5] F. S. Liddy, Natural language processing", The journal of department of information science, Page 1-20. (1999)"
[6] T. Masaru,"An Efficient All-paths Parsing Algorithm for Natural Languages", The journal of Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213, 25, page 1-36. (1984),
[7] J. Agar, The government machine: a revolutionary history of the computer. Cambridge, Massachusetts: MIT Press. ISBN 978-0-262-01202-7. (2003).
[8] P. McCorduck, Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1(2004),
[9] J. Weizenbaum, "ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine", Communications of the ACM **9** (1): 36–45, (January 1966) doi:10.1145/365153.365168
[10] T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", MIT AI Technical Report 235, February 1971
[11] La Fontaine, B., "Lasers and Moore's Law", SPIE Professional, Oct. 2010, p. 20; http://spie.org/x42152.xm