**[Context-Free Grammar (NLP Based)]**

***[Ghalib Hussain, 22K-4536, Sec: A]***

***[Wali Mohsin, 22K-4550, Sec: A]***

***[Rayyan Alam, 22K-4373, Sec: A]***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**0. Acknowledgment**

Most of the project has been covered under the programming concepts of Data Structures taught in the classes. Although, to increase the interactivity, and efficiency at some points in the program, few built-in/ helper functions were used from the internet. No specific website has been favored in our searches- the one with the best idea has been adopted. These websites include stackoverflow, geeksforgeeks, and tutorialspoint, and Wikipedia (to grasp the concept of context free grammar and structure of English Language and NLP). The ideas taken from internet are: Regex pattern- used in verification of email, and password mask function. We would also like to so thank our instructor Sir Ghufran for helping us understand the concept of data structures.

**1. Introduction**

The project implements basic CFG (Context-Free Grammar), a formal grammar in which every production rule is formed. CFG's are used to describe the structure and formations of sentences, words, and phrases in natural language. They support the natural language of linguists and form the structure of programming language, but our project has been restricted to the natural language of linguistics. In our project, English language statements have been parsed. Words will be matched from the corpus (collection of already defined stories of that norm) from open source dictionaries or URLs to check if the terms and sentences fall on proper grammar such as "I prefer a morning walk" is a valid structure whereas "Boys cricket play" is invalid. Our project follows the program's supervised learning approach as a general concept because the project judges sentences on sample inputs and some predefined or consistent rules fed in texts to compare the entered text.

The project we implemented uses the parsing technique to parse the sentence into words and store them into a "Queue" data structure. The use of the "QUEUE" data structure helps us keep comments in a FIFO order, and once the whole sentence has been parsed, each word's type is checked through the list of corpora of the English language through the filing technique. Once the class is determined, each class is stored with the word again using a "QUEUE." Later on, the predefined functions of a sentence are determined – the structure is checked to see if the sentence is a good structure or not using nested procedure and recursion. (Sentence structure diagram mentioned below) i.e. studying NLP, PP, VP, NN, etc.

The structure of the project is broken into a more straightforward form through the use of recursion, which uses an implicit stack.

The project has been limited to the students of Fast only. This has been done by encompassing login and signup functions with the help of "BST." This has been used to make sure that no one except for NU students can access the software, and to make it easier for the user, the user information is being stored in a text file so that the user may log in back whenever they want to check their sentence's structure. All of their knowledge, including NU id and password, has been restored.

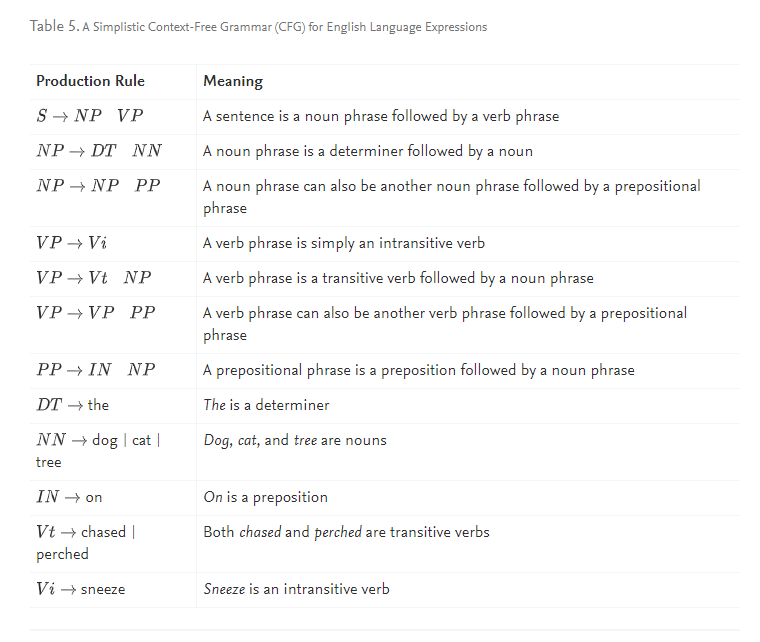
**2. Tools and technologies used**

Visual Studio Code, Dev C++ (Language Standard ISO C++ 11)

**Libraries, and functions (pre-defined):**

* + - 1. #include <iostream>, used for cin,cout (input/output stream) to take inputs and outputs.
      2. #include <fstream> , used for filing
      3. #include <cstring>, used for string primitive functions
      4. #include <windows.h> used for system("cls”) to clear screen.
      5. gotoxy(x,y) -> use to move the cursor with their respective x and y positions.
      6. #include <regex>, used for regex\_match() to match patterns with input strings.

**3. Sentence Structure Diagram**



**4. List of Corpus**

Nouns: http://www.desiquintans.com/downloads/nounlist/nounlist.txt

Verbs: https://www.talkenglish.com/vocabulary/top-1000-verbs.aspx

Prepositions: https://www.thefreedictionary.com/List-of-prepositions.htm

Pronouns: http://www.esldesk.com/vocabulary/pronouns

Adjectives: https://www.paperrater.com/page/lists-of-adjectives

Determiners: https://www.ef.com/wwen/english-resources/english-grammar/determiners/

Conjunctions: http://www.marshall.k12.il.us/data/webcontent/245/file/realname/files/List-of-Conjunctions.pdf

**5. Link to source**

<https://github.com/ashmalvayani/Context-Free-Grammar->

**6. Future work**

Context-Free Grammar is a versatile and comprehensive formal grammar rule for automata with extensive applications; the part that we implied in our project was for English Grammar Checker, and regarding all the regulations related to the English language as NP, PP, VP, delimiters, etc. It checks if the sentence has an invalid word type / invalid sentence structure, or if it’s a good structure, it will show all the corpora along with the type of phrases it contains by parsing the sentences.

Now, in future, we plan to extend this concept of sentence parsing or grammar structure checker for not just English language, but can raise it to compiler level or hardware level, the corpora will be replaced with compiler’s rules and syntax and the user-input command will be parsed and will check for all the laws of the compiler and will allow a user to enter next command if the existing one follow’s the rules.

For eg: cout << “Hello world” << endl;

This is a valid command, whereas

cout >> “Hello world” << endl;

It is not a valid command, thus shouldn’t allow continuing.

Though many compilers already exist to check for the structure of commands, since C++ is a vast language and numerous things are getting added today, the compiler needs to accommodate the new changes more. This concept will help in the promotion of AI-enabled NLP software.