BBM497: Introduction to Natural Language Processing Lab. **(Due: 04/05/2020)**

Submission Assignment #3

Instructor: Asst. Prof. Dr. Burcu CAN, Necva BÖLÜCÜ *Name:* Deniz Zağlı, *Netid:* 21527668

- **Libraries:**

  Actually, I used only one library in this assignment. That library is also python's random library. I used this library to create a random sentence from the rules when producing random sentences. In this way, random sentences suitable for the grammar were created.

- **Data Structures:**

  I used two data structures in the dataset reading process. The first is the list, the second is the dictionary. While reading the lines, we made the necessary split operation and processed the necessary lines, so we have a list. Afterwards, I took the necessary data for the dictionary by using the lists. The reason I use the list to hold the data is that its operations are easy and convenient. I used this data for the dictionary to create a CFG rules dictionary from the data I hold. The reason I use a dictionary when creating CFG rules is because the access is fast and I can access it with a key instead of an index. At the stage of producing random sentences, I kept the sentences I produced with the list. The reason I keep it on the list is that there is little data and it is easy to add. Again, for the same reason, I used a list when creating a table in CYK algorithm. The list helped me add and extract data. I repeat that I used the list data structure due to the low number of data. If the number of data was too high, I would have to use numpy array.

- **Part 1: Language Generation with CFG:**

  In this party, I produced random sentences using CFG rules. I mentioned the code content in the comments in the code. I used recursive algorithm while producing random sentences. The start parameter is given and random selection is made from the rules that this parameter can produce. Then, random selection is made from the rules that can be produced by the new rules. This process continues until the term chosen at random is terminal. Finally, the space at the end of the sentence is deleted and the initial letter is enlarged. In this way, a random sentence is formed. Let's examine randomly generated sentences.

  Sentences whose initial parameter is "ROOT":

  - Is it true that every mouse pickled a pickle ?
  - That mouse pickled want from prefer under wanted in the delicious mouse on the sandwich !
  - Is it true that ate on every sandwich in this floor on the sandwich from the delicious pickle in that fine delicious fine pickle in ate in washed in that floor from wanted on that sandwich to is under pickled with this sandwich on need with wanted from that pickle with this pickle pickled prefer in like from the president from kissed ?

  Sentences whose initial parameter is "NP":

  - This old pickle
  - Kissed to the president
  - his sandwich under this president to pickled on prefer with that president on that floor on wanted in pickled on every pickle with every sandwich to want

  Sentences whose initial parameter is "Noun":

  - Fine sandwich
  - Old beautiful old pickle

– Mouse

Let's examine the sentences produced by the random sentence generator. The sentences have different lengths because the sentences are completely random. We can see this clearly when we examine the sentences with the initial parameter ROOT. These sentences are grammatically correct but completely random. Since there are also punctuation marks in ROOT rules, the sentences that are produced are finished with punctuation marks different from the others. Let's look at sentences with an initial parameter of NP. Since NP contains three different rules, namely the way of going, the sentences created differ even if they follow the NP rule. Although these sentences are suitable for the NP grammar, they are not sentences. When we come to the sentences whose initial parameter is Noun, the situation is a little different. Noun has both a rule containing non terminals and a word containing terminals. For this reason, we can either continue with the adjective noun path with Noun or choose a direct noun. This also appears in the examples. The situation with more than one adjective is very unlikely, since it is chosen completely randomly. As a result of my 100 times trial, I was able to observe this situation. You can see this situation in the second of the sentences produced with the Noun parameter. This possibility is realized by the realization of the "Adj Noun" rule two or more times. It is 0.028 that this possibility comes twice in a row.

- **Part 2: Parsing Sentences with CYK Parser:**

  While creating the CYK parser algorithm, "https://www.xarg.org/tools/cyk-algorithm/" made use of the interactive example available on this site. I first created a table. I filled the first row of the table I created with the non-terminal equivalents of the given sentence. Then I observed a situation in the interactive example. The cells of the table with the rules to be combined advance first from bottom to top and second to the right bottom cross. If I encode this iteration and merge the rules in those cells of the table, and create another parent rule, I wrote the upper rule in the current cell. This iteration continued until the last element of the table. If the last element is S, the sentence is a grammatically correct sentence. If the last element is not S, it means either not a sentence or it is not grammatically correct. Detailed explanation of the algorithm and table sample are available in the comment lines in the code. Let's test my algorithm for two sentences we created for Part 1:

  – Is it true that every mouse pickled a pickle ?

  The output of the CYK algorithm is "Grammatically Correct". The table formed as a result of the CYK algorithm is as follows:

  [[['Det'], ['Noun'], ['Verb', 'NP'], ['Det'], ['Noun']], [['NP'], [], [], ['NP']], [[], [], ['VP']], [[], []], [['S']]]

  Let's look at another example:

  – Old beautiful old pickle ?

  The output of the CYK algorithm is "Grammatically Not Correct". The table formed as a result of the CYK algorithm is as follows:

  [[['Adj'], ['Adj'], ['Adj'], ['Noun']], [[], [], ['Noun']], [[], ['Noun']], [['Noun']]]

  In fact, this sentence is derived by the noun rule. However, the algorithm outputs the grammatically wrong output because there is no sentence according to the desired rules, that is, the last element of the table cannot be "S".