BBM497: Introduction to Natural Language Processing Lab. **(Due: 04/05/20)**

Submission Assignment #3

Instructor: Burcu CAN *Name:* Utku İPEK, *Netid:* 21627356

Abstract

This assignment consists of two tasks. The first task is to generate random sentences from given CFG rules, and the second task is to implement a CYK parser which determines whether the given random sentence is in the language or not. The details of my implementation can be found in below sections.

1 Language Generation with CFG

In this part of the assignment, first, I've implemented the *rules()* function which takes the folder path contains CFG rules as the argument, then returns the rule set as a list. Then, I've implemented another function to classify these rules of three category: non-terminal rules, terminal rules, and root rules. Non-terminal rules contains one non-terminal or two non-terminals on the right-hand side, terminal rules contains one terminal on the right-hand side, and root rules contain "ROOT" on the left-hand side. I've created dictionaries for all three of these rule types as the left-hand sides are the keys and the right-hand sides are the values. After that, I've implemented a function to reverse those dictionaries, which means as the right-hand sides of the rules are keys and the left-hand sides are the values. The reason is that when generating random sentences from rules or parsing the sentence with CYK algorithm, sometimes it is more efficient to use the reverse version of the dictionaries.

When I am done with the data structures to hold the rules, I've implemented the *randsentence()* function which uses the rule dictionaries and generate random sentences. This function also writes those sentences to an output file. An important point here is that the sentences can be generated either just using the vocabulary (terminals), or using the non-terminal rules. My implementation applies both of them. First, I've generated the sentences only using the vocabulary, and I've seen that the probability of generating a sentence that is accepted in the language by randomly choosing words is too low. After that I've generated the sentences from non-terminal rules using a recursive function I've implemented an I've seen that by choosing from the rules randomly, there is more chance to generate a sentence that is accepted in the language. The probability is still low but, compared to first one, it makes more sense to use the second method. Some of the generated sentences with both methods can be found in the next section.

2 Parsing Sentences with CYK Parser

In the second part of the assignment, I've implemented the CYK parser algorithm which is an instance of dynamic programming. This algorithm traverses an upper-triangular matrix from left to right and each column is filled from bottom to up (as shown in Figure 1) to determine whether the given sentence is accepted in the language or not. To represent this upper-triangular matrix, I've used a dictionary which holds the indices of the matrix as its keys. The reason I've used a dictionary as my parse table is efficiency in terms of space. By using a dictionary, I've just kept the necessary indices and the space hasn't wasted.

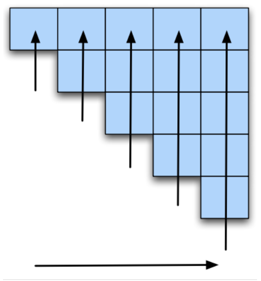

Figure 1: Upper Triangular Matrix (Figure is from https://web.stanford.edu/~jurafsky/slp3/13.pdf)

The upper-triangular matrix I've used as my parse table is a portion of an *(n+1) by (n+1)* matrix where n is the length of the sentence. In each loop, the algorithm parses the sentence in different ways, traverses the cells, an puts the rules accordingly. Finally, if the cell *(0,n)* contains the rule which defines a sentence, that means the given sentence is accepted in the language and my implementation is true. An important point here is that some of the grammar rules of this assignment are not in Chomsky Normal Form (CNF). Normally a basic CYK parser algorithm is for CNF grammars. For example there is a rule like *Noun Pronoun* which contains single non-terminal in the right-hand side. To handle this situation and handle the root rules situation, I've added extra parts to my CYK implementation.

After I've implemented the CYK parser, to apply it all the sentences I've generated, I've implemented a function. There is also another function to show the parse table of a given sentence. Below, some of the generated sentences and their results are shown below.

Sentences and generated by choosing randomly from vocabulary and their results:

- old on on kissed floor ! – False
- beautiful it a old the . – False
- beautiful you i want you ! – False
- is it true that mouse every this kissed pickled ? – False
- that pickle ! – False

Sentences generated by choosing randomly from rules and their results:

- that sandwich kissed every president . – True
- every floor washed me with a sandwich from that pickle on it in a floor ! – True
- a mouse ! – False
- this floor on you prefer every sandwich . – True
- is it true that beautiful floor ? – False

3 Conclusion

As seen in the previous section, generating random sentences from rules gives better results in the CYK parser than generating random sentences from vocabulary. Also, this assignment shows the importance of the CYK algorithm. It is a helpful tool for grammatically analyzing the sentences of a language. The CYK parser I've implemented is a recognizer. It can be transformed to a parser that draws the parse tree of the given sentence with some improvements in the future.