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

Submission Assignment #3

Instructor: Necva BÖLÜCÜ *Name:* Ahmet Faruk BAYRAK, *Netid:* 21426716

1 Introduction

In this assignment, we were expected to implement a **CYK parser** and a random sentence generator using **CFG rule set**

1.1 Context-free Grammar (CFG)

Context-free grammars (CFGs) are used to describe context-free languages. A context- free grammar is a set of recursive rules used to generate patterns of strings. A context- free grammar can describe all regular languages and more, but they cannot describe all possible languages.

1.2 Dataset

For creating the CFG rule set we used a grammar file (cfg.gr) which is provided for us in the Chomsky Normal Form (CNF) Example of this grammar file:

```
# Creating a sentence:

ROOT    S .
ROOT    S !
ROOT    is it true that S ?

#    S  = sentence
#    NP = noun phrase
#    VP = verb phrase
#    PP = prepositional phrase
#    Det = determiner
#    Prep = preposition
#    Adj = adjective
#    Pronoun  = Pronoun


S   NP VP
VP  Verb NP
NP  Det Noun
NP  Pronoun
NP  NP PP
PP  Prep NP
Noun    Adj Noun
```

```
# Vocabulary. There

Verb    ate
Verb    wanted
Verb    kissed
Verb    washed
Verb    pickled
Verb    is
Verb    prefer
Verb    like
Verb    need
Verb    want

Det the
Det a
Det every
Det this
Det that

Noun    president
Noun    sandwich
Noun    pickle
Noun    mouse
Noun    floor
```

2 Data structures

2.1 Creating The CFG Rule Set

First of all we need to create CFG rule set according to given grammar file. For this purpose I used dictionary to keep all grammar rules (non-terminals and terminals). My dictionary structure:

```
{'Adj': [['fine'], ['delicious'], ['beautiful'], ['old']], 'Det': [['the'], ['a'], ['every'], ['this'], ['that']], 'Pronoun': [['me'], ['i'], ['you'], ['it']],
 'Prep': [['with'], ['on'], ['under'], ['in'], ['to'], ['from']], 'ROOT': [['.', 'S'], ['!', 'S'], ['?', 'S', 'is it true that']], 'Noun': [['Adj', 'Noun'],
 ['president'], ['sandwich'], ['pickle'], ['mouse'], ['floor']], 'S': [['NP', 'VP']], 'NP': [['Det', 'Noun'], ['Pronoun'], ['NP', 'PP']], 'VP': [['NP', 'Verb']],
 'Verb': [['ate'], ['wanted'], ['kissed'], ['washed'], ['pickled'], ['is'], ['prefer'], ['like'], ['need'], ['want']], 'PP': [['NP', 'Prep']]}
```

2.2 Task 1: Language Generation with CFG

In this part we were expected to generate random sentences using grammer rules that we created from CFG file.

For this purpose I created a function which calls itself recursively until a sentence is created. In this function I used dept first approach. This function starts from **"ROOT"** and it will go down rule by rule until the terminal word. After that it adds this word to the list and it goes one step back and do the same thing for this

rule until reach final rule's terminal word.

This function creates grammatically correct sentences always. But the tricky part is there is a sentence boundary which we give. For example size of generated grammatically correct sentence is 20 and our sentence boundary is 15. For this situation function will take first 15 words so sentence will be grammatically incorrect probably. The sentence with the first 15 words might be grammatically correct as well but the probability is small.

According to my output results: If generated sentence has a punctuation in the end (it means it reaches final rule such as "ROOT: S ."), this sentence is grammatically correct always. But If there is no punctuation in the end of the sentence it means that this sentence is probably grammatically incorrect.

Also there is one more analysis of my results: If the sentence boundary number is high, number of generated grammatically correct sentence is very high as well since they could reach the final rule. But If the sentence boundary number is low, number of generating grammatically correct sentences will be less since they could not reach the final rule (they were still in the middle of rule tree).

When the number of sentence boundary is **5**:

```
Sentence: Is it true that that
Grammatically situation: INCORRECT.
**************
Sentence: Every pickle to the pickle
Grammatically situation: INCORRECT.
**************
Sentence: Is it true that this
Grammatically situation: INCORRECT.
**************
Sentence: It ate that mouse under
Grammatically situation: INCORRECT.
**************
Sentence: I like you!
Grammatically situation: CORRECT.
```

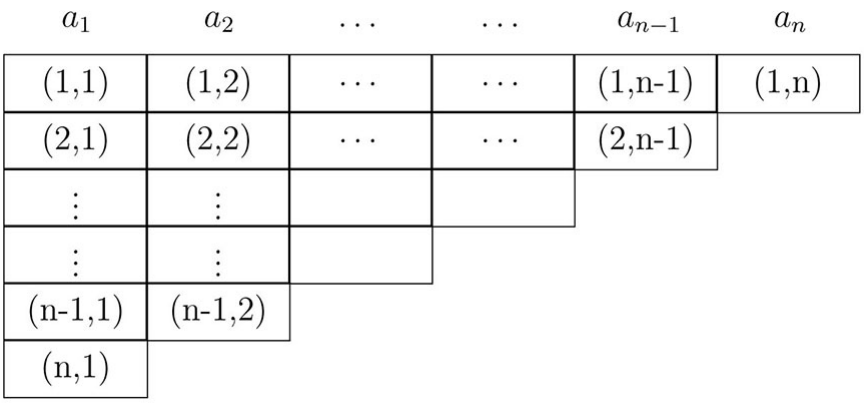
When the number of sentence boundary is **20**:

```
Sentence: Is it true that a fine pickle need it?
Grammatically situation: CORRECT.
**************
Sentence: The sandwich want it to every delicious pickle from you on it under i on i with every president with
Grammatically situation: INCORRECT.
**************
Sentence: Is it true that you in the mouse like this pickle on me?
Grammatically situation: CORRECT.
**************
Sentence: Is it true that every mouse pickled it?
Grammatically situation: CORRECT.
**************
Sentence: You wanted a pickle!
Grammatically situation: CORRECT.
```

2.3 Parsing Sentences with CYK Parser

In this part we were expected to implement CYK parser as a recognizer which tells whether a given sentence is grammatically correct or not according to the same CFG rule set.

For this purpose I used triangular table which is matrix actually. In this matrix first row is unigram of words, second row is bigram of words and it goes until the sentence itself.



In my function, first of all I initialize the matrix table which is a list. After that I start with first line which is unigram of words. For only first row I had one special search: after finding non-terminal of terminal word in the rule set, I checked whether this non-terminal word is in the rights side of any rule as well. For example for word **"you"** I checked grammar rule set and I found this word can only be generated from **"Pronoun"** and I took Pronoun. After that I checked for Pronoun and I found that **"NP: Pronoun"** rule so I took **NP** as well. I had to do this for only first row to get a correct result.

After filling the first row I started to fill the rest of the table. For filling the rest of the cells I look for all possible pairs according to past results. For example in the bigram cells I looked for unigram of two words and did the **cartesian product** to find all possible pairs. After determining the all possible pairs I check whether these pairs in the right side of rule set. If they are I took the left side of rule and so on.

When filling the table is done, I start to check for whether this sentence is grammatically correct. If the root cell has a "S" it means this sentence is grammatically correct.

```
Sentence: This delicious sandwich in me wanted the mouse under you in that floor!

Table:

['Det']['Adj']['Noun']['Prep']['Pronoun', 'NP']['Verb']['Det']['Noun']['Prep']['Pronoun', 'NP']['Prep']['Det']['Noun']
[]['Noun'][]['PP']['VP'][]['NP'][]['PP']['PP'][]['NP']
['NP'][][][][]['VP']['PP'][][][]['PP']
['PP'][][][]['S'][]['NP'][][]['NP']
['NP'][][][][][]['VP']['PP'][]['PP']
['VP'][][][][]['S'][][][]
[][][][][][][]['NP']
['S'][][][][][]['VP']
[][][][][]['S']
['S'][][][][]
[][][][]
[][][]
['S']

Grammatically situation: CORRECT.
```

```
Sentence: I ate i with it to i to you to

Table:

['Pronoun', 'NP']['Verb']['Pronoun', 'NP']['Prep']['Pronoun', 'NP']['Prep']['Pronoun', 'NP']['Prep']['Pronoun', 'NP']['Prep']
['VP']['VP']['PP']['PP']['PP']['PP']['PP']['PP']['PP']
['S'][]['NP'][]['NP'][]['NP'][]
[]['VP']['PP']['PP']['PP']['PP']['PP']
['S'][]['NP'][]['NP'][]
[]['VP']['PP']['PP']['PP']
['S'][]['NP'][]
[]['VP']['PP']
['S'][]
[]

Grammatically situation: INCORRECT.
```