Assignment 3

Due on May 04, 2019 (23:59:59)

Click here to accept your Assignment 3

Introduction

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.

0.1 Part 1: Language Generation with CFG

In this first part of the assignment, you will implement a random sentence generator. For this, a sample CFG rule set is provided for you in the Chomsky Normal Form (CNF). Using the grammar rules defined in a grammar file, you will generate a sentence randomly.

The format of the CFG rules is as follows:

```
S               NP   VP             #S -> NP VP
Noun         book                  #Noun -> book
```

Every sentence begins with a root which is defined as follows:

```
ROOT S              #ROOT -> S
```

Each nonterminal can have multiple expansions. For example, NP either can be expanded as Det Noun or Det Adj Noun. You should choose one of the rules randomly while you generate a sentence. Run your program multiple times and observe what kind of sentences it tends to generate. Discuss it in your report. (The CFG rules)

0.2 Part 2: Parsing Sentences with CYK Parser

In the second part of the assignment, you will implement CYK parser as a recognizer which tells whether a given sentence is grammatically correct or not according to the same CFG rule set (used in the first part of the assignment).

Note: No out-of-vocabulary (OOV) word will be included in the given sentence.

To this end, you will implement a simple CYK class with the following interfaces:

- **rules():** It takes only one argument: folder path and returns CFG rule set.
- **randsentence():** It takes CFG rule set and name of the output file and returns generated sentences (Also they will be written to the output file) .

- **CYKParser():** It takes a sentence and print whether this sentence is grammatically correct or not.

Submit

You are required to submit all your code. You will implement the assignment in **Python** (Python 3.5). You will submit a report in latex format template). The codes you will submit should be well commented. Your report should be self-contained and should contain a brief overview of the problem and the details of your implemented solution. Give the answers of all questions raised in the definition of the assignment above. You can include pseudocode or figures to highlight or clarify certain aspects of your solution.

- report.pdf
- code/ (directory containing all your codes as Python file .py)

Grading

- Code (90 points) : Task 1: 35 points, Task 2 : 55 points
- Report (10 points)

Note: Preparing a good report is important as well as the correctness of your solutions! You should write your results for each part of the task and answer of the questions related with parts.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.