CSCE 638

Programming Assignment 3

Name: Sun Yul Lee

ID: 824007724

● Objective

The objective of this assignment is to write a program that implements the classic CKY algorithm for syntactic parsing. The programs train a Naïve Bayes classifier and a Perceptron classifier on the imdb1 data set. The programs need to classify an entire movie review as positive or negative.

● Implementation

To implement CKY algorithm, I created a class CKY and following functions. In the class constructor, I added two lists for holding a set of non-terminal symbols and a asset of terminal symbols.

```
class CKY:
  def __init__(self):
    """CKY algorithm initialization"""
    self.non_terminals = []
    self.sigma = []
```

Figure 1. Added instance variables of a class CKY

- def readRules(self, fileName)

  This function reads rule text file to create dictionary of grammar rules for CKY syntactic parsing algorithm. For each line in the text file, the function separates a line into non-terminal symbol, the symbol's derivation, and a probability of the rule. Then it sets non-terminal symbol as the dictionary's key and sets a list of derivation and a probability of the rule as the dictionary's value. The function also creates a set of non-terminal symbols and a set of terminal symbols. The function returns the dictionary of grammar rules.

- def readSentences(self, fileName)

  This function reads sentences text file to create a list of sentences. For each line in the text file, the function appends a line as an element to a list of sentences. It returns the list of sentences.

- def parsing(self, rules, sentence)

  This function performs the CKY algorithm. First, the function creates a table of π values and a table of back pointers. Then it initializes cells in the table of π values where a row index and a column index are the same. For the other cells in the table, this function calls a function max_pi() to calculate a π value and get back pointer arguments of the cells.

- def max_pi(self, i, j, rules, pis)

  This function calculates the equation in Figure 2. It returns the calculated value and back pointer arguments π value

$$\max_{\substack{X \to Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \to Y Z) \times \pi(i, s, Y) \times \pi(s+1, j, Z) \right)$$

Figure 2. Formula for calculating π value

- def writeParsing(self, pis, pbs, sentence)

  This function writes outputs of CKY syntactic parsing algorithm to a text file. It uses a table of π values and a table of back pointers to write the outputs. The format of the outputs is the same as the format in the file “sents_parsing_output.txt”.

- def parseDir(grammarDir, sentencesDir)

  This function creates object for CKY algorithm. Using the object, it read grammar rules and sentences from text files and perform the syntactic parsing on the sentences.

Compile and Execution

To execute the program, user needs to type “python CKY.py <grammar_file> <sentences_file>” in a command line prompt where <grammar_file> is a text file which contains grammar rules with probabilities in the Chomsky normal form and <sentences_file> is a text file which contains a set of sentences.

```
(base) C:₩Users₩etional₩Desktop₩2020₩spring₩CSCE638₩pa3-cky₩pa3-cky>python CKY.py grammar_rules.txt sents.txt
```

Figure 3. Executing the CKY algorithm in my device

Result

After the execution of the program, a text file “output.txt” is created. The text file contains the output of the syntactic parsing with CKY algorithm in a format from the example output file “sents_parsing_output.txt”. With the given set of sentences, although the order is not the same, the output is the same as the output in “sents_parsing_output.txt” except line 40 in “output.txt”. The detail of this difference is explained in a section “Problem”.

```
PROCESSING SENTENCE people tanks fish
SPAN:  people
P(NP) = 0.3
P(N) = 0.5
P(V) = 0.1

SPAN:  tanks
P(NP) = 0.1
P(N) = 0.2
P(V) = 0.3

SPAN:  fish
P(VP) = 0.1
P(NP) = 0.2
P(N) = 0.2
P(V) = 0.6

SPAN:  people tanks
P(VP) = 0.005 (BackPointer = (1,V,NP))
P(NP) = 0.003 (BackPointer = (1,NP,NP))

SPAN:  tanks fish
P(S) = 0.01 (BackPointer = (2,NP,VP))
P(VP) = 0.03 (BackPointer = (2,V,NP))
P(NP) = 0.002 (BackPointer = (2,NP,NP))

SPAN:  people tanks fish
P(S) = 0.009 (BackPointer = (1,NP,VP))
P(VP) = 0.0001 (BackPointer = (1,V,NP))
P(NP) = 6e-05 (BackPointer = (1,NP,NP))
```

Figure 4. Example of the output (Input sentence: people tanks fish)

Problem

In the output of the program with the given set of sentences, line 40 in “output.txt” is different from the output in “sents_parsing_output.txt”. The reason that the output of the program is different from the given output is floating point error in python. A text “fish people fish” can be parsed into “fish” and “people fish”, and “fish people” and “fish” with the same rule, NP -> NP NP. The π value of the first case is 0.1 * 0.2 * 0.006 = 0.00012 and the π value of the second case is 0.1 * 0.006 * 0.2 = 0.00012. Since the π values are the same, with my implementation, the back pointer table should remember the arguments in the first case. However, because of floating point error in python, the π value of the second case calculated in the program is slightly bigger than the π value of the first case and the back pointer table remembers the arguments in the second case.

```
SPAN:  fish people fish
P(S) = 0.002 (BackPointer = (1,NP,VP))
P(VP) = 0.0018 (BackPointer = (1,V,NP))
P(NP) = 0.00012 (BackPointer = (2,NP,NP))
```

Figure 5. Output in “output.txt”

```
SPAN:  fish people fish
P(VP) = 0.0018 (BackPointer = (1,V,NP))
P(NP) = 0.00012 (BackPointer = (1,NP,NP))
P(S) = 0.002 (BackPointer = (1,NP,VP))
```

Figure 6. Output in “sents_parsing_output.txt”

Limitation

When the program reads grammar rules file, it assumes that the grammar rules are in Chomsky Normal Form. Also, it assumes that a non-terminal and its derivations are separated by two spaces. If there are two derivations, derivations are separated by one space. A grammar rule and its probability are separated by at least one-tab space. Therefore, if grammar rules are not in this format, error will occur and the program will not work. For a sentences file, the program assumes that each line in the file is one sentence.