Assignment 2
NLP 202: Natural Language Processing II

University of California Santa Cruz

Winter 2023

This assignment is to be done in Python 3.

Your final writeup for this assignment, including the problem and the report of the experimental findings of the programming part, should be no more than four pages long. You should submit it as a PDF. We *strongly* recommend typesetting your scientific writing using LaTeX. Some free tools that might help: TexStudio (Windows), MacTex (Mac), TexMaker (cross-platform), and Overleaf (online).

Instructions

In this assignment, you will implement the basic CKY algorithm and the weighted CKY algorithm. For each programming problems, please describe your experimental procedures clearly.

1 Filling Out The Chart By Hand (25%)

In this section, given the grammar below, fill out the chart assuming the use of the CKY algorithm. Include pointers to sub-constituents. If there is more than one way to build a particular non-terminal in a cell (for example more than one way to create an S), list the duplicates with different pointers to the sub-constituents.

S $\rightarrow$ NP VP	NP $\rightarrow$ British	NP $\rightarrow$ waffles
NP $\rightarrow$ JJ NP	JJ $\rightarrow$ British	VP $\rightarrow$ waffles
VP $\rightarrow$ VP NP	NP $\rightarrow$ left	P $\rightarrow$ on
VP $\rightarrow$ VP PP	VP $\rightarrow$ left	NP $\rightarrow$ Falklands
PP $\rightarrow$ P NP		

British	left	waffles	on	Falklands
$[0, 1]$				
	$[1, 2]$			
		$[2, 3]$		
			$[3, 4]$	
				$[4, 5]$

2 Programming: CKY Algorithm (25%)

In this section, you will do a similar thing as Part 1 but using your implemented CKY algorithm. Here is the CKY algorithm (from the reading):

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
          table[j − 1, j] ← table[j − 1, j] ∪ A
    for i ← from j − 2 downto 0 do
      for k ← i + 1 to j − 1 do
        for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
              table[i,j] ← table[i,j] ∪ A
```

Figure 13.5 The CKY algorithm.

However, different from part 1, please use a **set** for storing the non-terminals in each cell, and **do not** count duplicates as separate entries. This is the usual way of implementing the CKY algorithm, and is what the algorithm printed here does. Allowing duplicate entries for each non-terminal would make the runtime exponential, because there could be an exponential number of entries in the top cell (one for each parse).

Please print your final parse table. You can use python `tabulate` package[1] to make your output more readable.

3 Programming: Weighted CKY Algorithm (25%)

A **Probabilistic Context-Free Grammar** (PCFG) is exactly like a context-free grammar, but it has a *probability* component associated with it. The definition: a PCFG consists of:

- A set N of NON-TERMINAL symbols;
- A set Σ of TERMINAL symbols;
- A START SYMBOL $\mathsf{S} \in N$;
- A set R of RULES, such that each rule $r \in R$ has the form $X \to \alpha$, where $X \in N$ and $\alpha \in (N \cup \Sigma)^*$—that is, the LEFT-HAND SIDE of r is a single symbol in N, and the RIGHT-HAND SIDE of r is a (possibly empty) sequence of symbols drawn from N and/or Σ (the * is the Kleene star that we saw in regular expressions)
- A PROBABILITY FUNCTION P that maps each rule $r \in R$ to a probability ranging from $[0, 1]$, such that for every non-terminal $X \in N$,

$$\sum_{\alpha:\ (X \to \alpha) \in R} P(X \to \alpha) = 1$$

  That is, for every non-terminal X in the grammar, the probabilities of the rules rewriting X sum to 1. You can think of $P(X \to \alpha)$ as indicating the *conditional probability* of daughters α appearing in the tree, given that X is the mother.

In this section, given a simple PCFG (in CNF form) below, modify your implemented CKY algorithm to parse the sentence "**astronomers saw stars with ears**".

[1] https://pypi.org/project/tabulate/

S	$\rightarrow$ NP VP	1.0	NP	$\rightarrow$ NP PP	0.4
PP	$\rightarrow$ P NP	1.0	NP	$\rightarrow$ astronomers	0.4
VP	$\rightarrow$ V NP	0.7	NP	$\rightarrow$ ears	0.18
VP	$\rightarrow$ VP PP	0.3	NP	$\rightarrow$ saw	0.04
P	$\rightarrow$ with	1.0	NP	$\rightarrow$ stars	0.18
V	$\rightarrow$ saw	1.0	NP	$\rightarrow$ telescopes	0.1

Please print your final parse table, the most probable parse tree and its probability.

4 Programming: Sum Over All Parse Trees (25%)

In this section, given the same PCFG as Part 3, modify your implemented CKY algorithm to marginalize over the trees to get the probability of the sentence.

Submission Instructions

Submit a single zip file (`A2.zip`) on Canvas. The zip file should contain the following:

- **Code**: You will submit your code together with a neatly written README file to instruct how to run your code with different settings. We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade. However, please provide well commented code if you want partial credit. If you have multiple files, please provide a short description in the preamble of each file.
- **Report**: In the writeup, be sure to fully describe your models and experimental procedure. Provide graphs, tables, charts or other summary evidence to support any claims you make.As noted above, your writeup should be four pages long, or less, in PDF (one-inch margins, reasonable font sizes). Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.