

DATA 641 - HW3

Name: Precious Worgu

Student ID: 119343890

Problem 1a: CKY parsing table for the sentence "She eats a fish with a fork"

```
In [1]: import numpy as np

def cky_parse(sentence, grammar):
    # Split sentence into words
    words = sentence.split()
    n = len(words)

    # Initialize parse table
    table = np.empty((n, n), dtype=object)
    for i in range(n):
        for j in range(n):
            table[i, j] = set()

    # Fill in diagonal entries
    for i in range(n):
        for rule in grammar:
            if words[i] in rule[1]:
                table[i, i].add(rule[0])

    # Fill in upper-triangle entries
    for j in range(1, n):
        for i in range(j-1, -1, -1):
            for k in range(i, j):
                for rule in grammar:
                    if len(rule[1]) == 2 and rule[1][0] in table[i, k] and rule[1][1] in table[k+1, j]:
                        table[i, j].add(rule[0])

    # Return the parse table
    return table

# Example usage
grammar = [
    ('S', ('NP', 'VP')),
    ('PP', ('P', 'NP')),
    ('NP', ('Det', 'N')),
    ('NP', ('N', 'N')),
    ('NP', ('She',)),
    ('VP', ('V', 'NP')),
    ('VP', ('VP', 'PP')),
    ('VP', ('eats',)),
    ('Det', ('a',)),
    ('N', ('fish',)),
    ('N', ('fork',)),
    ('N', ('a',)),
    ('P', ('with',)),
    ('V', ('eats',))
]

sentence = "She eats a fish with a fork"
table = cky_parse(sentence, grammar)
print(table)
```

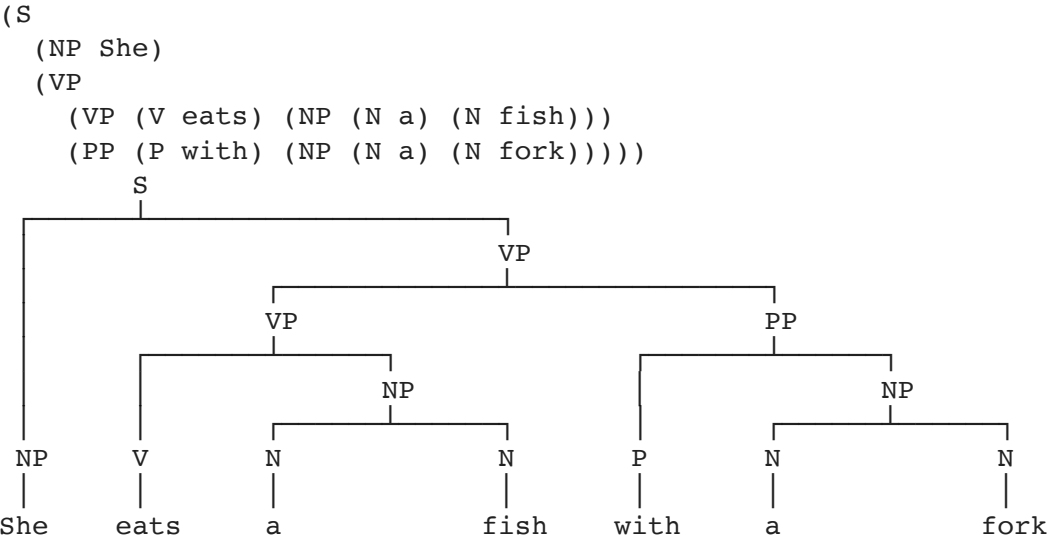
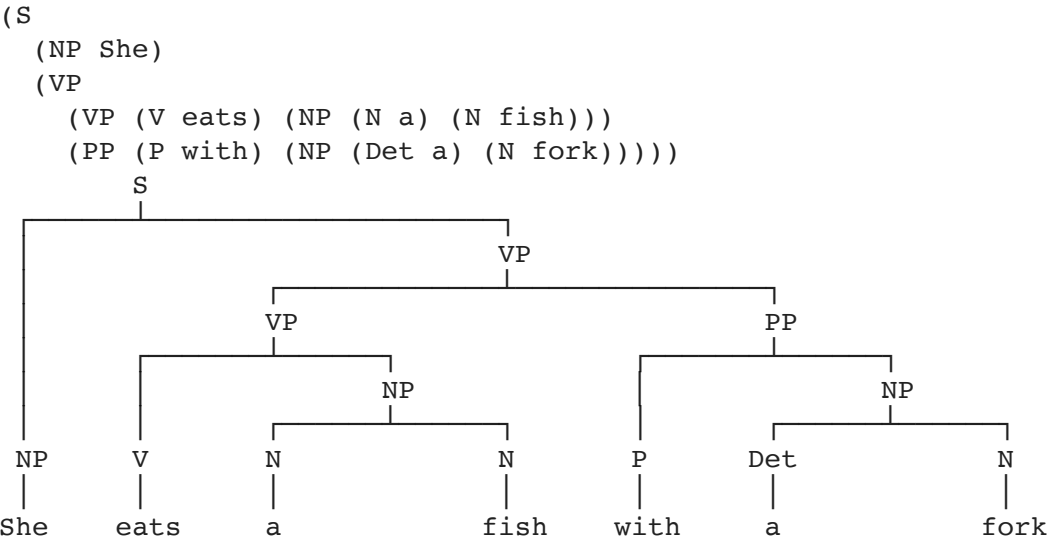
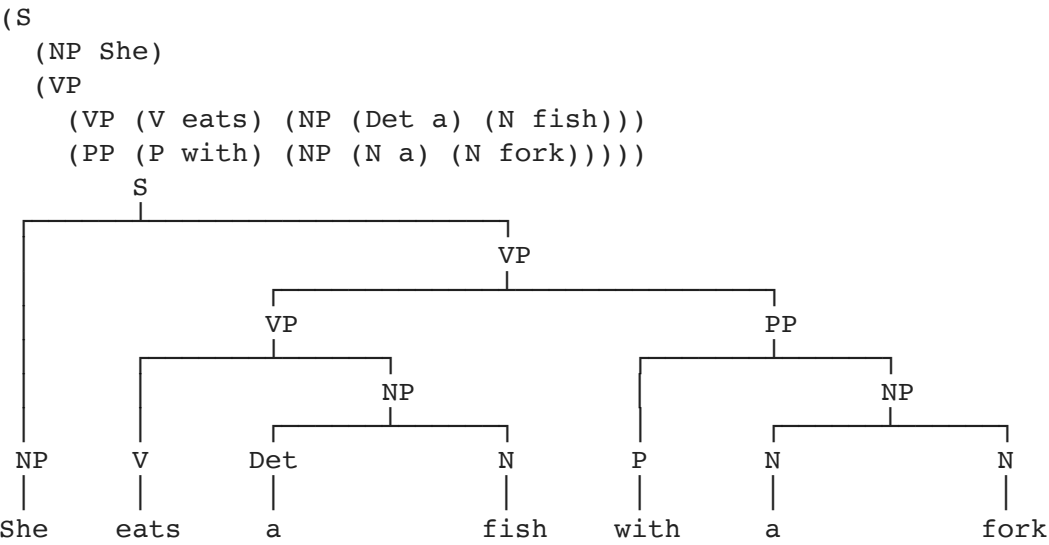
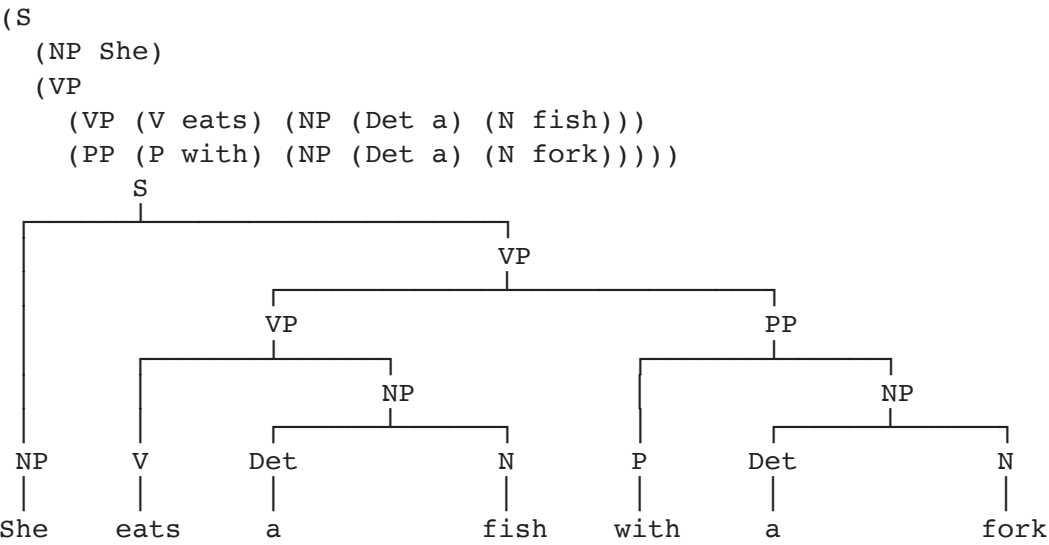
```
[[{'NP'} {'S'} set() {'S'} set() set() {'S'}]
 [set() {'V', 'VP'} set() {'VP'} set() set() {'VP'}]
 [set() set() {'Det', 'N', 'VP'} {'NP'} set() set() set()]
 [set() set() set() {'N'} set() set() set()]
 [set() set() set() set() {'P'} set() {'PP'}]
 [set() set() set() set() set() {'Det', 'N', 'VP'} {'NP'}]
 [set() set() set() set() set() set() {'N'}]]
```

1b) Parse trees for the sentence "She eats a fish with a fork"

```
In [2]: import nltk

grammar1 = nltk.CFG.fromstring("""
S -> NP VP
VP -> V NP | VP PP | "eats"
PP -> P NP
V -> "eats"
NP -> "She" | Det N | N N
Det -> "a"
N -> "fish" | "fork" | "a"
P -> "with"
""")

sent = "She eats a fish with a fork".split()
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(sent):
    print(tree)
    tree.pretty_print(unicodelines=True, nodedist=4)
```



Problem 2: Syntactic dependency trees

```
In [3]: import spacy
from nltk import Tree

en_nlp = spacy.load('en_core_web_sm')

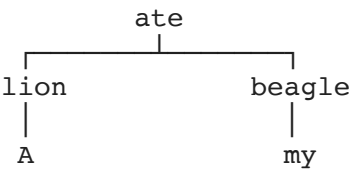
doc = en_nlp("A lion ate my beagle")

def to_nltk_tree(node):
    if node.n_lefts + node.n_rights > 0:
        return Tree(node.orth_, [to_nltk_tree(child) for child in node.children])
    else:
        return node.orth_

print(f"{'Token':{8}} {'dependence':{6}} {'head text':{9}} {'Dependency explained'} ")
for token in doc:
    print(f"{'token.text':{8}} {'token.dep_+' =>':{10}} {'token.head.text':{9}} {spacy.explain(token.dep_)} ")

[to_nltk_tree(sent.root).pretty_print(unicodelines=True, nodedist=4) for sent in doc.sents]
```

Token	dependence	head text	Dependency explained
A	det =>	lion	determiner
lion	nsubj =>	ate	nominal subject
ate	ROOT =>	ate	root
my	poss =>	beagle	possession modifier
beagle	dobj =>	ate	direct object



Out[3]: [None]

```
In [4]: en_nlp = spacy.load('en_core_web_sm')

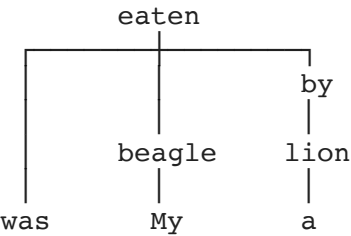
doc = en_nlp("My beagle was eaten by a lion")

def to_nltk_tree(node):
    if node.n_lefts + node.n_rights > 0:
        return Tree(node.orth_, [to_nltk_tree(child) for child in node.children])
    else:
        return node.orth_

print(f"{'Token':{8}} {'dependence':{6}} {'head text':{9}} {'Dependency explained'} ")
for token in doc:
    print(f"{'token.text':{8}} {'token.dep_+' =>':{10}} {'token.head.text':{9}} {spacy.explain(token.dep_)} ")

[to_nltk_tree(sent.root).pretty_print(unicodelines=True, nodedist=4) for sent in doc.sents]
```

Token	dependence	head text	Dependency explained
My	poss =>	beagle	possession modifier
beagle	nsubjpass =>	eaten	nominal subject (passive)
was	auxpass =>	eaten	auxiliary (passive)
eaten	ROOT =>	eaten	root
by	agent =>	eaten	agent
a	det =>	lion	determiner
lion	pobj =>	by	object of preposition



Out[4]: [None]

```
In [5]: en_nlp = spacy.load('en_core_web_sm')

doc = en_nlp("The beagle was eager to eat")

def to_nltk_tree(node):
    if node.n_lefts + node.n_rights > 0:
        return Tree(node.orth_, [to_nltk_tree(child) for child in node.children])
    else:
        return node.orth_

print(f"{'Token':{8}} {'dependence':{6}} {'head text':{9}} {'Dependency explained'} ")
for token in doc:
    print(f"{'token.text':{8}} {'token.dep_+' =>':{10}} {'token.head.text':{9}} {'spacy.explain(token.dep_)} ")

[to_nltk_tree(sent.root).pretty_print(unicodelines=True, nodedist=4) for sent in doc.sents]
```

Token	dependence	head text	Dependency explained
The	det =>	beagle	determiner
beagle	nsubj =>	was	nominal subject
was	ROOT =>	was	root
eager	acomp =>	was	adjectival complement
to	aux =>	eat	auxiliary
eat	xcomp =>	eager	open clausal complement

was

beagle

eager

beagle

eat

The

to

Out[5]: [None]

```
In [6]: en_nlp = spacy.load('en_core_web_sm')

doc = en_nlp("The beagle was easy to eat")

def to_nltk_tree(node):
    if node.n_lefts + node.n_rights > 0:
        return Tree(node.orth_, [to_nltk_tree(child) for child in node.children])
    else:
        return node.orth_

print(f"{'Token':{8}} {'dependence':{6}} {'head text':{9}} {'Dependency explained'} ")
for token in doc:
    print(f"{'token.text':{8}} {'token.dep_+' =>':{10}} {'token.head.text':{9}} {'spacy.explain(token.dep_)} ")

[to_nltk_tree(sent.root).pretty_print(unicodelines=True, nodedist=4) for sent in doc.sents]
```

Token	dependence	head text	Dependency explained
The	det =>	beagle	determiner
beagle	nsubj =>	was	nominal subject
was	ROOT =>	was	root
easy	acomp =>	was	adjectival complement
to	aux =>	eat	auxiliary
eat	xcomp =>	easy	open clausal complement

was

beagle

easy

beagle

eat

The

to

Out[6]: [None]