Question 1

Rank 1: the most similar pair; Rank 15: the least similar pair

Rank the pairs based on how similar I think the two senses in each pair are:

Rank	Word 1	Word 2
1	puppy#n#1	dog#n#1
2	reptile#n#1	snake#n#1
3	neuron#n#1	cell#n#2
4	jazz#n#2	music#n#1
5	knowledge#n#1	learning#n#1
6	water#n#1	solvent#n#1
7	young#a#1	age#n#1
8	penguin#n#1	south#n#4
9	jelly#n#1	jellyfish#n#2
10	book#n#1	dictatorship#n#1
11	hunt#v#1	bird#n#1
12	artificial#a#3	fake#a#2
13	journey#v#1	trip#n#1
14	photo#n#1	capture#v#1
15	monarchy#n#1	pizza#n#1

Rank the pairs according to similarity measures of lesk:

Rank	Word 1	Word 2	lesk
1	neuron#n#1	cell#n#2	772
2	young#a#1	age#n#1	748
3	puppy#n#1	dog#n#1	687
4	reptile#n#1	snake#n#1	190
5	water#n#1	solvent#n#1	142
6	jazz#n#2	music#n#1	108
7	knowledge#n#1	learning#n#1	89
8	hunt#v#1	bird#n#1	70
9	journey#v#1	trip#n#1	51
10	book#n#1	dictatorship#n#1	45
11	monarchy#n#1	pizza#n#1	20
12	penguin#n#1	south#n#4	17

13	photo#n#1	capture#v#1	16
14	jelly#n#1	jellyfish#n#2	9
15	artificial#a#3	fake#a#2	5

Rank the pairs according to similarity measures of jcn:

Rank	Word 1	Word 2	jcn
1	reptile#n#1	snake#n#1	20.996
2	puppy#n#1	dog#n#1	0.3001
3	neuron#n#1	cell#n#2	0.2239
4	knowledge#n#1	learning#n#1	0.1961
5	water#n#1	solvent#n#1	0.1356
6	book#n#1	dictatorship#n#1	0.0596
7	jazz#n#2	music#n#1	0.0000
8	monarchy#n#1	pizza#n#1	0.0000
9	penguin#n#1	south#n#4	0.0000
10	jelly#n#1	jellyfish#n#2	0.0000
11	young#a#1	age#n#1	-1
12	hunt#v#1	bird#n#1	-1
13	journey#v#1	trip#n#1	-1
14	photo#n#1	capture#v#1	-1
15	artificial#a#3	fake#a#2	-1

Rank the pairs according to similarity measures of path:

Index	Word 1	Word 2	path
1	puppy#n#1	dog#n#1	0.5
2	reptile#n#1	snake#n#1	0.3333
3	neuron#n#1	cell#n#2	0.3333
4	knowledge#n#1	learning#n#1	0.25
5	jazz#n#2	music#n#1	0.25
6	water#n#1	solvent#n#1	0.1429
7	monarchy#n#1	pizza#n#1	0.0714
8	penguin#n#1	south#n#4	0.0714
9	jelly#n#1	jellyfish#n#2	0.0667

10	book#n#1	dictatorship#n#1	0.0625
11	young#a#1	age#n#1	-1
12	hunt#v#1	bird#n#1	-1
13	journey#v#1	trip#n#1	-1
14	photo#n#1	capture#v#1	-1
15	artificial#a#3	fake#a#2	-1

- Yes, different measures do produce different rankings since they use different factors to compute the similarity of each synset pair.

- The measure path is the most consistent with my own ranking.

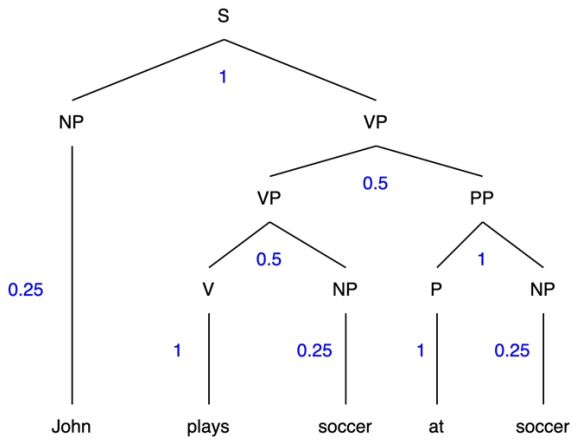
- As observed from the tables above, the measure path and jcn produce many "-1", which is a weakness, since these two measures only accept [n, n] and [v, v] POS pairs. They do not support [adj, adj] POS pairs, (e.g., the artificial#a#3 and fake#a#2 pair), and many other pairs (e.g., [n, v]). Most of the inconsistencies in the rankings compared with my own ranking are those pairs with a measure value of -1. Since the values are all -1, their similarity rankings cannot be distinguished from each other and those pairs are always ranked at the end, which cannot reflect their similarity at all compared with those with a measure greater than 0.

- There is a possible error in the ranking produced by jcn similarity measure. Because of the display limitation (only display 4 significant figures after the decimal point), many values appear to be 0.0000 but in reality, they are just very close to 0 but not equal to 0. In this case, the ranking of those pair with value 0.0000 cannot be distinguished from each other even if their similarity values produce by jcn might be different. In this case, the ranking might not be accurate.

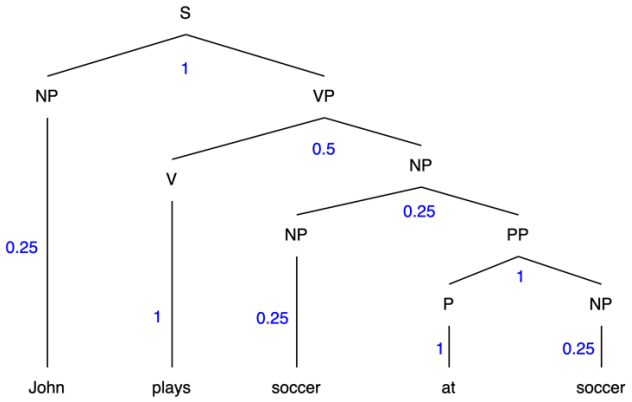
Question 2

(a) The provided algorithm does give the right answer on the sample input *pcfg-grammar.pl*.
There are two ways parsing the sentence "John plays soccer at soccer".

We have the tree A (probability labeled in blue):



and the tree B (probability labeled in blue):



Hence, we have:
$P(Tree^A) = 1\times0.25\times0.5\times0.5\times1\times0.25\times1\times1\times0.25 = 0.00390625$
$P(Tree^B) = 1\times0.25\times0.5\times1\times0.25\times0.25\times1\times1\times0.25 = 0.001953125$

Tree A has a higher probability, and the provided algorithm chooses the tree A with higher probability.

Question 2 (b)

In [1]:
```
from nltk import Tree
from pprint import pprint
```

In [2]:
```
allSentences = []
allTrees = []
with open('HW4/silly-corpus') as corpus:
    for setence in corpus:
        # remove the "/" around each word
        clearSentence = setence.rstrip().replace("/", "")
        allSentences.append(clearSentence)
        allTrees.append(Tree.fromstring(clearSentence))
```

In [3]:
```
nonTerminalDict = {}
ruleDict = {}
```

In [4]:
```
def updateNonTerminalDict(label):
    if label not in nonTerminalDict:
        # create a new key: value pair with key = label, value = 1
        nonTerminalDict[label] = 1
    else:
        nonTerminalDict[label] += 1
    return;
```

In [5]:
```
def updateRuleDict(rule):
    if rule not in ruleDict:
        # create a new key: value pair key = rule, value = 1
        ruleDict[rule] = 1
    else:
        ruleDict[rule] += 1
    return;
```

In [6]:
```
def parse(tree):
    stringifyNodes = ""
    for node in tree:
        if type(node) is not str:
            stringifyNodes = stringifyNodes + str(node.label()) + " "
            # parse subtree
            parse(node)
        else:
            # add the child node and padding
            stringifyNodes = stringifyNodes + node + " "

    tLabel = tree.label()
    stringifyNodes =  " ".join(stringifyNodes.split())
    rule = tLabel, stringifyNodes

    updateNonTerminalDict(tLabel)
    updateRuleDict(rule)
    return;
```

In [7]:
```
# parse all trees
# and update nonTerminalDict and ruleDict with all trees data
for tree in allTrees:
    parse(tree)
```

In [8]:
```
pprint(nonTerminalDict)
```

```
{'NP': 25, 'P': 4, 'PP': 4, 'S': 10, 'V': 10, 'VP': 13}
```

In [9]:
```
pprint(ruleDict)
```

```
{('NP', 'John'): 10,
 ('NP', 'NP PP'): 1,
 ('NP', 'school'): 4,
 ('NP', 'soccer'): 10,
 ('P', 'at'): 4,
 ('PP', 'P NP'): 4,
 ('S', 'NP VP'): 10,
 ('V', 'plays'): 10,
 ('VP', 'V NP'): 10,
 ('VP', 'VP PP'): 3}
```

Compute estimates for the grammar probabilities based on silly-corpus

In [10]:
```
probabilities = {}

for rule in ruleDict:
    nonTermNum = nonTerminalDict[rule[0]]
    ruleNum = ruleDict[rule]
    prob = ruleNum/nonTermNum

    prettyRule = rule[0] + " -> " + rule[1]

    probabilities[prettyRule] = prob
```

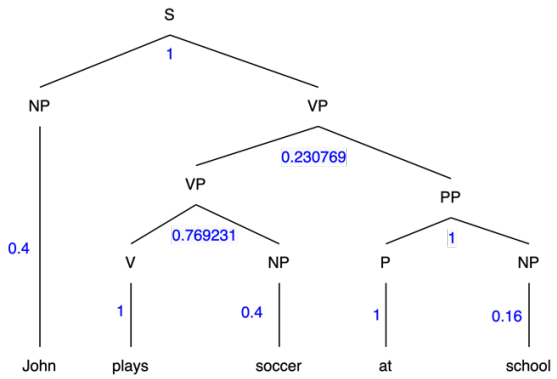
Based on silly-corpus, the probability for each rule is:

In [11]:
```
pprint(probabilities)
```

```
{'NP -> John': 0.4,
 'NP -> NP PP': 0.04,
 'NP -> school': 0.16,
 'NP -> soccer': 0.4,
 'P -> at': 1.0,
 'PP -> P NP': 1.0,
 'S -> NP VP': 1.0,
 'V -> plays': 1.0,
 'VP -> V NP': 0.7692307692307693,
 'VP -> VP PP': 0.23076923076923078}
```

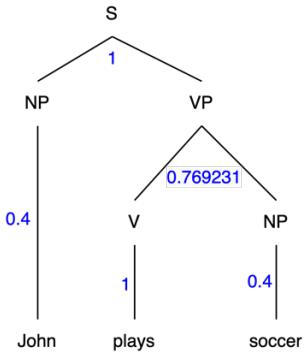
(b)

- Given new grammar, the probability of the most likely parse:
  - for the sentence “John plays soccer at school”:
    Tree C:



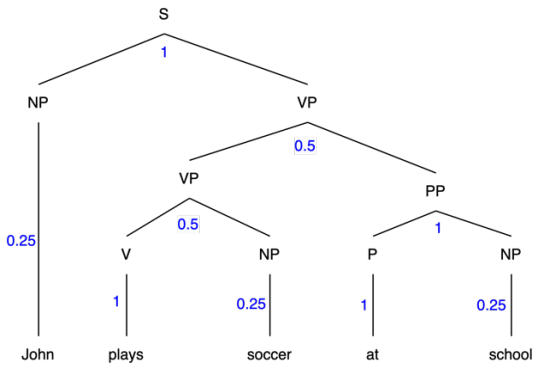
$P(Tree^C) = 1\times0.4\times0.230769\times0.769231\times1\times0.4\times 1 \times1\times0.16 = 0.004544$

  - for the sentence “John plays soccer”:
    Tree D:



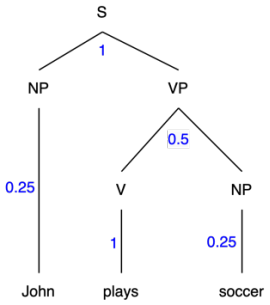
$P(Tree^D) = 1\times0.4\times0.769231\times1\times0.4 = 0.123077$

- Using the old grammar, the probability of the most likely parse:
  - for the sentence “John plays soccer at school”:
    Tree E:



$P(Tree^E) = 1\times0.25\times0.5\times0.5\times1\times0.25\times1\times1\times0.25 = 0.00390625$

  - for the sentence “John plays soccer”:
    Tree F:



$P(Tree^F) = 1\times0.25\times0.5\times1\times0.25 = 0.03125$

- The results from the old and new grammar are different for both sentences.

For the sentence “John plays soccer at school”: the probabilities of 1) VP-> VP PP; 2) VP->V NP; 3) NP-> terminal word are different in the old and the new grammar. Given the silly-corpus, it is more likely for VP to be expanded into V and NP than into VP and PP. In addition, words like “John” and “soccer” are more common than “school” in grammar, and thus the probability of NP being expanded into “John” or “soccer” is higher than into “school”. The new grammar takes these cases into account, but the old grammar, which assigns uniformly distributed probabilities to rules for each non-terminal, does not. The same reason applies to the sentence “John plays soccer” for explaining why the new grammar and old grammar produce different probabilities.