1. (24 pts) For each recursive transition network (RTN) below, write a grammar that accepts exactly the same language as the RTN.

(a) RTN #1:
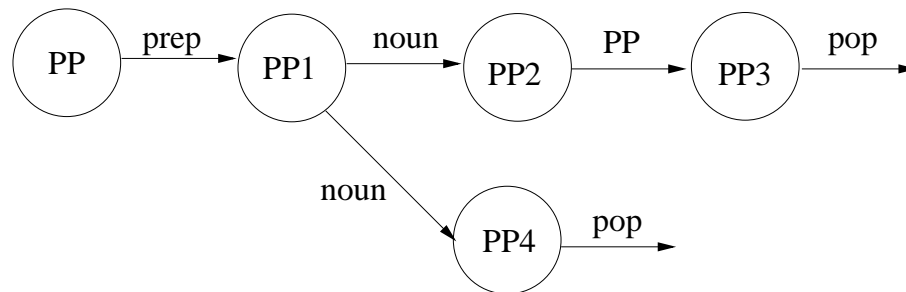
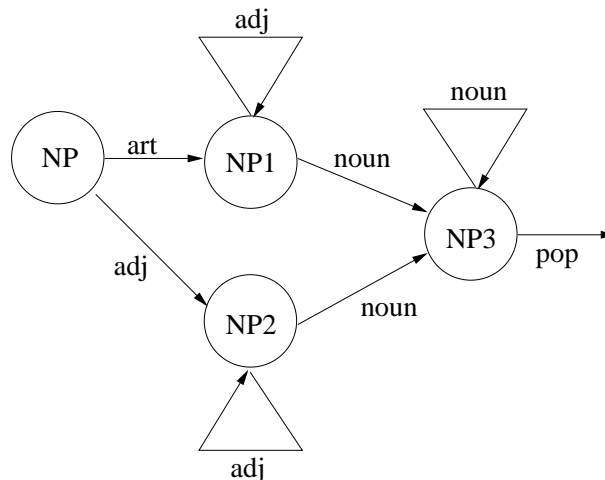PP → prep noun PP
PP → prep noun

(b) RTN #2:

NP → art NP1
NP → adj NP2
NP1 → adj NP1
NP2 → adj NP2
NP1 → NP3
NP2 → NP3
NP3 → noun NP3
NP3 → noun

If we were to write it more generically like the example which Professor worked out in class, it would be

NP → art X
NP → adj X
X → adj X
X → Y
Y → noun Y
Y → noun

(c) RTN #3:
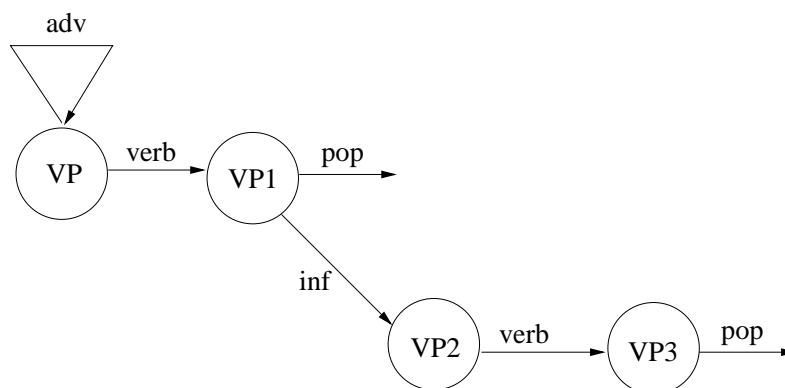
VP → adv VP
VP → verb inf verb
VP → verb

---

2. (30 pts) Consider the grammar and lexicon below:

Grammar	Lexicon
S → NP VP	*trust* : noun, verb
NP → noun	*shrinks* : noun, verb
NP → noun noun	
VP → verb	
VP → verb NP	

List all of the entries that would be put on the chart during **bottom-up chart parsing** for the sentence **'trust shrinks'**. Each chart entry should be a constituent or a rule, with a start and end position indicating which words have been matched by the constituent or rule.

To get you started, the list below contains chart entries for the part-of-speech tag constituents for the words 'trust' and 'shrinks' and for two rules that should be on the chart. Your job is to complete this list by adding the rest of the constituents and rules that would be put on the chart during parsing.

For the rules, please use an asterisk (*) to separate the components of the rule that have been matched from the ones that have not yet been matched. For example, the rule "NP → noun * noun [1-2]" means that the first noun in this rule has been successfully matched with a constituent starting in position 1 and ending in position 2.

Complete the Chart Entries for Bottom-up Chart Parsing of 'Trust Shrinks'

Constituent/Rule	Start-End
noun("trust")	[1-2]
verb("trust")	[1-2]
noun("shrinks")	[2-3]
verb("shrinks")	[2-3]
NP → noun *	[1-2]
NP	[1-2]
NP → noun * noun	[1-2]
VP → verb *	[1-2]
VP	[1-2]
VP → verb * NP	[1-2]
S → NP *VP	[1-2]
NP → noun *	[2-3]
NP	[2-3]
NP → noun noun * (carried over rule gets completed here)	[1-3]
NP	[1-3]
NP → noun * noun	[2-3]
VP → verb NP *(carried over rule gets completed here)	[1-3]
VP	[1-3]
VP → verb *	[2-3]
VP	[2-3]
VP → verb * NP	[2-3]
S → NP VP *	[1-3]
S	[1-3]

3. (16 pts) Consider the following tongue twister:

```
whether the weather is warm , whether the weather is hot , we have to
put up with the weather , whether we like it or not .
```

Using the tongue twister above as your text corpus, compute the following probabilities. Note that this text sample contains 28 words because each comma and period counts as a word. *Please leave your answers in fractional form with the frequency counts used for the computation! For example, we want to see answers of the form 14/28, not .50*

- P("or") = $\frac{1}{28}$

- P("weather") = $\frac{3}{28}$

- P("weather" | "the") = $\frac{3}{3}$ = 1

- P("whether" | "we") = $\frac{0}{2}$ = 0

- P("warm" | "is") = $\frac{1}{2}$

3

- $P(\text{"hot"} \mid \text{"weather"}, \text{"is"}) = \frac{1}{2}$

- $P(\text{"is"} \mid \text{"the"}, \text{"weather"}) = \frac{2}{3}$

- $P(\text{"up"} \mid \text{"to"}, \text{"put"}) = \frac{1}{1} = 1$

4. (18 pts) Consider the sentence:
   `John Smith planned to fish for silver trout and char but he forgot to bring the fishing gear`

   Suppose your part-of-speech (POS) tagger assigns these POS tags to the sentence:

   `John/NOUN Smith/NOUN planned/VERB to/PREP fish/NOUN for/PREP silver/NOUN trout/NOUN and/CONJ char/VERB but/CONJ he/PRO forgot/VERB to/INF bring/VERB the/ART fishing/VERB gear/NOUN`

   Assume that the **correct** POS tags for the sentence are:

   `John/NOUN Smith/NOUN planned/VERB to/INF fish/VERB for/PREP silver/ADJ trout/NOUN and/CONJ char/NOUN but/CONJ he/PRO forgot/VERB to/INF bring/VERB the/ART fishing/GER gear/NOUN`

   Based on the information above, answer the questions below. Please leave your answers in fractional form!

   (a) What is the overall accuracy of your POS tagger?
   $\frac{13}{18}$

   (b) What is the recall of your POS tagger for verbs?
   $\frac{3}{4}$

   (c) What is the precision of your POS tagger for verbs?
   $\frac{3}{5}$

   (d) What is the recall of your POS tagger for nouns?
   $\frac{4}{5}$

   (e) What is the precision of your POS tagger for nouns?
   $\frac{4}{6}$

   (f) What is the recall of your POS tagger for prepositions (PREP tag)?
   $\frac{1}{1}$

   (g) What is the precision of your POS tagger for prepositions (PREP tag)?
   $\frac{1}{2}$

   (h) Using the **correct** POS tags, what is the lexical generation probability P(planned | VERB)?
   $\frac{1}{4}$

   (i) Using the **correct** POS tags, what is P(VERB | planned), which means the probability that the word "planned" should have the tag VERB?
   $\frac{1}{1}$

5. (12 pts) Consider the probability that the sentence *"A witty saying proves nothing"* (a quote from Voltaire) should be assigned the part-of-speech (POS) tag sequence *"ART ADJ NOUN VERB NOUN"*.

$$P(ART\ ADJ\ NOUN\ VERB\ NOUN \mid A\ witty\ saying\ proves\ nothing)$$

If you perform statistical part-of-speech tagging with N-gram language models to compute this probability, show the equation that would be used for each type of N-gram language model below. You must instantiate the equation with the specific words and part-of-speech tags shown above, but no numbers are needed.

If necessary, use the symbol ϕ_0 to denote the beginning of a sentence (i.e., the position immediately to the left of the first word) and the symbol ϕ_{-1} to denote two positions to the left of the first word.

(a) Show the equation using a unigram language model.

$$= \prod_{i=1}^{n} P(T_i) * P(w_i|T_i)$$

$=$[P(A|ART) * P(ART)]* [P(witty|ADJ) * P(ADJ)] * [P(saying|NOUN) * P(NOUN)] * [P(proves|VERB) * P(VERB)] * [P(nothing|NOUN) * P(NOUN)]

(b) Show the equation using a bigram language model.

$$= \prod_{i=1}^{n} P(T_i|T_{i-1}) * P(w_i|T_i)$$
$=$ [P(ART | ϕ) * P(A|ART)] * [P(ADJ|ART) * P(witty|ADJ)] * [P(NOUN|ADJ) * P(saying|NOUN)] * [P(VERB|NOUN) * P(proves|VERB)] * [P(NOUN|VERB) * P(nothing|NOUN)]

(c) Show the equation using a trigram language model. Each trigram should be written as $P(Z \mid X\ Y)$, where X precedes Y in the input.

$$= \prod_{i=1}^{n} P(T_i|T_{i-2}, T_{i-1}) * P(w_i|T_i)$$

$=$[P(ART | ϕ_{i-1}, ϕ_i) * P(A|ART)] * [P(ADJ|ϕ_i,ART) * P(witty|ADJ)] *[P(NOUN| ART, ADJ) * P(saying|NOUN)] * [P(VERB|ADJ, NOUN) * P(proves|VERB)] * [P(NOUN|NOUN, VERB) * P(nothing|NOUN)]

Question #6 is for CS-6340 students ONLY!

6. (18 pts) Consider the grammar and lexicon below:

Grammar	Lexicon
S → NP VP	*trust* : noun, verb
NP → noun	*shrinks* : noun, verb
NP → noun noun	
VP → verb	
VP → verb NP	

5

List all of the entries that would be put on the chart during **top-down chart parsing** for the sentence **'trust shrinks'**. Each chart entry should be a constituent or a rule, with a start and end position indicating which words have been matched by the constituent or rule.

To get you started, the list below contains chart entries for the part-of-speech tag constituents for the words 'trust' and 'shrinks' and for three rules that should be on the chart. Your job is to complete this list by adding the rest of the constituents and rules that would be put on the chart during parsing.

For the rules, please use an asterisk (*) to separate the components of the rule that have been matched from the ones that have not yet been matched. For example, the rule "NP → * noun noun [1-1]" means that no nouns in this rule have been matched yet but the rule can begin matching constituents starting in position 1.

Complete the Chart Entries for Top-down Chart Parsing of 'Trust Shrinks'

Constituent/Rule	Start-End
noun("trust")	[1-2]
verb("trust")	[1-2]
noun("shrinks")	[2-3]
verb("shrinks")	[2-3]
S → * NP VP	[1-1]
NP → * noun	[1-1]
NP → * noun noun	[1-1]
NP → noun *	[1-2]
NP	[1-2]
NP → noun * noun	[1-2]
S → NP * VP	[1-2]
VP → * verb	[2-2]
VP → * verb NP	[2-2]
VP → verb *	[2-3]
VP	[2-3]
VP → verb * NP	[2-3]
S → NP VP *	[1-3]
S	[1-3]