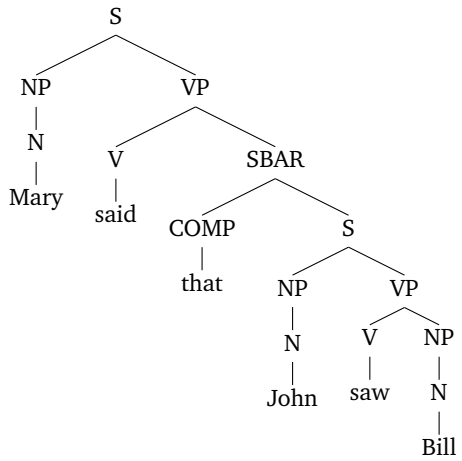
# Advanced Human Language Technologies
# Exercises on Parsing

## Dependency Parsing

### Exercise 1.

Given the sentence *Mary said that John saw Bill*, with the following parse tree:



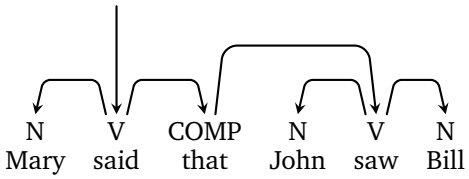

And the following grammar rules (where the superscript $+$ indicates the head):

$$\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}^{+}$$
$$\mathrm{NP} \rightarrow \mathrm{N}$$
$$\mathrm{VP} \rightarrow \mathrm{V}^{+}\ \mathrm{NP}$$
$$\mathrm{VP} \rightarrow \mathrm{V}^{+}\ \mathrm{SBAR}$$
$$\mathrm{SBAR} \rightarrow \mathrm{COMP}^{+}\ \mathrm{S}$$

1. List the headwords of the following non-terminals:
   - the SBAR
   - the topmost S
   - the VP "*said that John saw Bill*"
2. Draw the dependency tree resulting from the conversion using the given head rules.

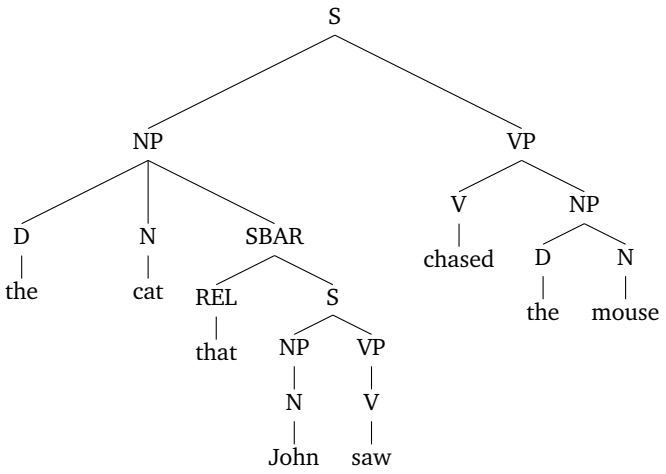
### SOLUTION

1. Headwords of the requested non-terminals are:
   - the SBAR $\longrightarrow$ *that*
   - the topmost S $\longrightarrow$ *said*
   - the VP "*said that John saw Bill*" $\longrightarrow$ *said*
2. The dependency tree resulting from the conversion using the given head rules is:


N V COMP N V N
Mary said that John saw Bill

## Exercise 2.

Given the sentence *The cat that John saw chased the mouse*, with the following parse tree:



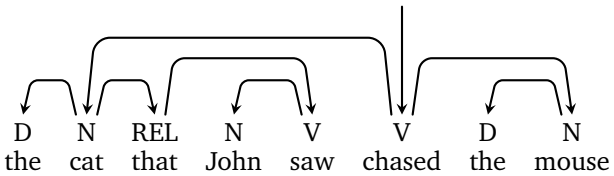

And the following grammar rules (where the superscript $+$ indicates the head):

$$\begin{aligned}
&\mathrm{S} \rightarrow \mathrm{NP\ VP}^{+}\\
&\mathrm{NP} \rightarrow \mathrm{N}\\
&\mathrm{NP} \rightarrow \mathrm{D\ N}^{+}\\
&\mathrm{NP} \rightarrow \mathrm{D\ N}^{+}\ \mathrm{SBAR}\\
&\mathrm{VP} \rightarrow \mathrm{V}^{+}\ \mathrm{NP}\\
&\mathrm{VP} \rightarrow \mathrm{V}\\
&\mathrm{SBAR} \rightarrow \mathrm{REL}^{+}\ \mathrm{S}
\end{aligned}$$

1. List the headwords of the following non-terminals:
   - the SBAR
   - the NP "*The cat that John saw*"
   - the topmost S
   - the VP "*chased the mouse*"
2. Draw the dependency tree resulting from the conversion using the given head rules.

## SOLUTION

1. Headwords of the requested non-terminals are:
   - the SBAR $\longrightarrow$ *that*
   - the NP "*the cat that John saw*" $\longrightarrow$ *cat*
   - topmost S $\longrightarrow$ *chased*
   - the VP "*chased the mouse*" $\longrightarrow$ *chased*
2. The dependency tree resulting from the conversion using the given head rules is:



## Exercise 3.

Consider the sentence: *John quit his job*.
Draw the following dependency parses.

a) (2,1), (0,2), (1,3), (3,4)

b) (2,1), (0,2), (2,3), (3,4)

c) (2,1), (0,2), (2,4), (3,4)

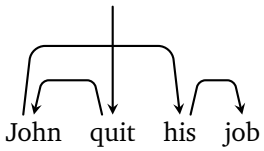
d) (2,1), (0,2), (2,4), (4,3)

e) (0,1), (1,2), (2,3), (3,4)

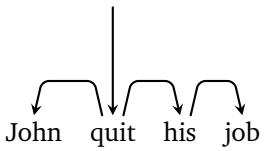
- Which are invalid parses and why?
- Which are projective parses?

## SOLUTION

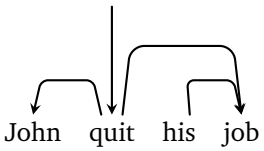
a) (2,1), (0,2), (1,3), (3,4)



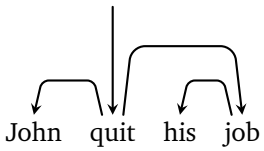

b) (2,1), (0,2), (2,3), (3,4)



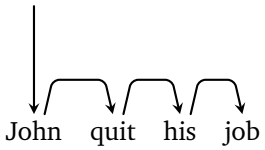

c) (2,1), (0,2), (2,4), (3,4)




d) (2,1), (0,2), (2,4), (4,3)




e) (0,1), (1,2), (2,3), (3,4)




- All parses are valid, except (c) which is not a tree since node *job* has two parents).
- Projective parses are (b), (d), and (e). Tree (a) is not projective since the root (0,2) arc crosses the (1,3) arc.

## Exercise 4.

In a global linear model for dependency parsing, the feacture vector $f(x, y)$ for any sentence $x$ paired with a dependency tree $y$ is defined as:

$$f(x, y) = \sum_{(h,m) \in y} \mathbf{f}(x, h, m)$$

where $\mathbf{f}(x, h, m)$ is a function that maps a dependency $(h, m)$ and a sentence $x$ to a local feature vector.

We want the vector $f(x, y)$ to have exactly two dimensions, each dimension having the following value:

$f_1(x, y)$ = num of times a dependency with head *car* and modifier *the* is seen in $(x, y)$

$f_2(x, y)$ = num of times a dependency with head part-of-speech NN, modifier part-of-speech DT, and no adjective (JJ) between the DT and the NN is seen in $(x, y)$

Assuming that each element in the sentence $x_i$ is a pair $(word, PoS)$, and that the functions $word(x_i)$ and $pos(x_i)$ return the value for each component of the pair:

1. Give a definition of the function $\mathbf{f}(x, h, m) = \langle \mathbf{f}_1(x, h, m), \mathbf{f}_2(x, h, m) \rangle$ that leads to the above definition of $f(x, y)$.
2. Compute the value of $f(x, y)$ for the following pair $(x, y)$:

   $x =$ *The/DT car/NN with/IN the/DT red/JJ hood/NN won/VBD the/DT car/NN race/NN*
   $y = \{(2, 1), (7, 2), (2, 3), (3, 6), (6, 4), (6, 5), (0, 7), (7, 10), (10, 8), (10, 9)\}$

## SOLUTION

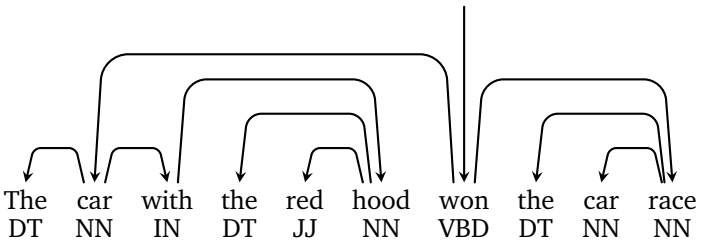
1. Since $\mathbf{f}(x, h, m) = \langle \mathbf{f}_1(x, h, m), \mathbf{f}_2(x, h, m) \rangle$, to define $\mathbf{f}$ we just need to define the indicator feature functions $\mathbf{f}_1$ and $\mathbf{f}_2$:

$$\mathbf{f}_1(x_{1:n}, h, m) = \begin{cases} 1 & \text{if } word(x_h) = \texttt{car} \text{ and } word(x_m) = \texttt{the} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_1(x_{1:n}, h, m) = \begin{cases} 1 & \text{if } pos(x_h) = \text{NN} \text{ and } pos(x_m) = \text{DT} \text{ and} \\ & \quad \nexists\ i : m < i < h \vee h < i < m : pos(x_i) = \text{JJ} \\ 0 & \text{otherwise} \end{cases}$$

2. We need to compute $\mathbf{f}_1(x, h, m)$ and $\mathbf{f}_1(x, h, m)$ for each arc in the tree, and then sum them to obtain the global vector $f(x, y)$.

   The given tree $y$ is:




   The features we get for each arc are:

| | |
|---|---|
| $\mathbf{f}_1(x, 2, 1) = 1$ | $\mathbf{f}_2(x, 2, 1) = 1$ |
| $\mathbf{f}_1(x, 7, 2) = 0$ | $\mathbf{f}_2(x, 7, 2) = 0$ |
| $\mathbf{f}_1(x, 2, 3) = 0$ | $\mathbf{f}_2(x, 2, 3) = 0$ |
| $\mathbf{f}_1(x, 3, 6) = 0$ | $\mathbf{f}_2(x, 3, 6) = 0$ |
| $\mathbf{f}_1(x, 6, 4) = 0^{(1)}$ | $\mathbf{f}_2(x, 6, 4) = 0^{(2)}$ |
| $\mathbf{f}_1(x, 6, 5) = 0$ | $\mathbf{f}_2(x, 6, 5) = 0$ |
| $\mathbf{f}_1(x, 0, 7) = 0$ | $\mathbf{f}_2(x, 0, 7) = 0$ |
| $\mathbf{f}_1(x, 7, 10) = 0$ | $\mathbf{f}_2(x, 7, 10) = 0$ |
| $\mathbf{f}_1(x, 10, 8) = 0^{(1)}$ | $\mathbf{f}_2(x, 10, 8) = 1$ |
| $\mathbf{f}_1(x, 10, 9) = 0$ | $\mathbf{f}_2(x, 10, 9) = 0$ |

(1) modifier is *the* but head is not *car*

(2) head is NN and modifier is DT but there is a JJ in between.

Finally, $f(x, y)$ is the sum of $\mathbf{f} = \langle \mathbf{f}_1, \mathbf{f}_2 \rangle$ over all arcs, thus $f(x, y) = (1, 2)$.

## Exercise 5.

*Recall the factored linear models for labeled dependency parsing. An arc-factored model computes:*

$$\begin{aligned} tree(x_{1:n}) &= \operatorname*{argmax}_{y \in \mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \mathbf{f}(x, y) \\ &= \operatorname*{argmax}_{y \in \mathcal{Y}(\mathbf{x})} \sum_{\langle h,m,l \rangle \in y} \mathbf{w} \cdot \mathbf{f}(x, h, m, l) \end{aligned} \quad (1)$$

*In the function, $x_{1:n}$ is an input sentence of $n$ tokens ($x_i$ is the $i$-th token). $\mathcal{Y}(x)$ is the set of all possible dependency trees for $x$ (each $y \in \mathcal{Y}(x)$ is a dependency tree). The tuple $\langle h, m, l \rangle$ is a labeled dependency: $h$ is is the index of the head word (we have $0 \leq h \leq n$, and $h = 0$ indicates the root token); $m$ is the index of the modifier word (we have $1 \leq m \leq n$), and $l$ is the syntactic label of that dependency (assume $\mathcal{L}$ is the set of possible syntactic relations (e.g. subject, object, modifier, etc.), and that $l \in \mathcal{L}$).*

In what follows, assume $pos(x_i)$ and $word(x_i)$ for $i \in \{1 \dots n\}$ return respectively the part-of-speech and word form in position $i$ in the sentence.

As usual we will define features using feature templates that capture certain syntactic properties. For example, an important property is to consider the compatibility of head-modifier relations with respect to part-of-speech tags. As a particular example, a verb will typically have nouns and adverbs as possible modifiers, but will never have determiners (since these modify nouns).

The following feature template will capture this information:

$$\mathbf{f}_{1,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \\ 0 & \text{otherwise} \end{cases}$$

In the template above $a$ and $b$ are possible PoS tags. Note that this template ignores the label. We could have another template that looks at PoS compatibility in conjunction with a label $c \in \mathcal{L}$:

$$\mathbf{f}_{2,a,b,c}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \text{ and } l = c \\ 0 & \text{otherwise} \end{cases}$$

1. Write feature templates that capture the following properties:

   (a) Lexical compatibility. For example, "boy" and "dog" are possible subject modifiers for the verb "eat", but "stone" or "pizza" are not likely subjects; on the other hand, "pizza" is a likely modifier for an object relation with "eat". Write two templates, one ignoring and the other considering the syntactic label:

   - $\mathbf{f}_{3,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$ and the modifier is $b$
   - $\mathbf{f}_{4,a,b,c}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the relation is $c$.

   (b) Adjectives in English appear before nouns ("small dog"), while for Spanish and Catalan they appear after nouns ("gos petit"). Write templates that capture the relative position of the modifier with respect to the head. Specifically, the features need to capture whether the modifier is to the left or to the right of the head, and whether the two words are adjacent or not. Write templates that only captures the relative position, and others that capture the relative position together with the pos tags or the words.

   - $\mathbf{f}_5(x_{1:n}, h, m, l)$ : The modifier is to the left of the head word.
   - $\mathbf{f}_6(x_{1:n}, h, m, l)$ : The modifier is to the right of the head word.
   - $\mathbf{f}_7(x_{1:n}, h, m, l)$ : The modifier is immediately left of the head word.
   - $\mathbf{f}_8(x_{1:n}, h, m, l)$ : The modifier is imediately right of the head word.
   - $\mathbf{f}_{9,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is to the left of the head word.
   - $\mathbf{f}_{10,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is to the right of the head word.

- $\mathbf{f}_{11,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is immediately left of the head word.
- $\mathbf{f}_{12,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is immediately right of the head word.
- $\mathbf{f}_{13,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is to the left of the head word.
- $\mathbf{f}_{14,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is to the right of the head word.
- $\mathbf{f}_{15,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is immediately left of the head word.
- $\mathbf{f}_{16,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is immediately right of the head word.

(c) In a noun phrase such as "many small hungry dogs" we expect to find a sequence of determiners and adjectives before a noun, and don't expect to find verbs in the middle of this sequence. Write feature templates that capture the pos tags of words that appear between the head and the modifier.

- $\mathbf{f}_{17,a}(x_{1:n}, h, m, l)$ : The PoS tag $a$ appears between the modifier and the head word.

(d) Write feature templates that capture Subject-Verb-Object phenomena[1] and variations (SOV, SVO, OVS, ...). Try to be general: assume a part of speech of a head word (e.g. verb) and two syntactic relations (e.g. subject and object), and write templates that can capture the relative position of the relations with respect to the head word. Illustrate the type of features that your templates can and can not capture.

- $\mathbf{f}_{18}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its left, and it is the subject.
- $\mathbf{f}_{19}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its right, and it is the subject.
- $\mathbf{f}_{20}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its left, and it is the object.
- $\mathbf{f}_{21}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its right, and it is the object.

2. Using the previous templates, compute the value of $f(x, y)$ for the following pair $(x, y)$:

$x =$ *the/DT big/JJ cat/NN eats/VBZ fresh/JJ fish/NN*
$y = \{(3, 1, \text{det}), (3, 2, \text{nmod}), (4, 3, \text{subj}), (0, 4, \text{root}), (6, 5, \text{nmod}), (4, 6, \text{obj})\}$

## SOLUTION

1. Feature patterns

(a) Lexical compatibility

- $\mathbf{f}_{3,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$ and the modifier is $b$

$$\mathbf{f}_{3,a,b,}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{4,a,b,c}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the relation label is $c$

$$\mathbf{f}_{3,a,b,c}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \text{ and } l = c \\ 0 & \text{otherwise} \end{cases}$$

(b) head-modifier order

- $\mathbf{f}_5(x_{1:n}, h, m, l)$ : The modifier is to the left of the head word.

$$\mathbf{f}_5(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } m < h \\ 0 & \text{otherwise} \end{cases}$$

[1] See `http://en.wikipedia.org/wiki/Subject-verb-object`

- $\mathbf{f}_6(x_{1:n}, h, m, l)$ : The modifier is to the right of the head word.

$$\mathbf{f}_6(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } m > h \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_7(x_{1:n}, h, m, l)$ : The modifier is immediately left of the head word.

$$\mathbf{f}_7(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } m = h - 1 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_8(x_{1:n}, h, m, l)$ : The modifier is imediately right of the head word.

$$\mathbf{f}_8(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } m = h + 1 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{9,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is to the left of the head word.

$$\mathbf{f}_{9,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \text{ and } m < h \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{10,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is to the right of the head word.

$$\mathbf{f}_{10,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \text{ and } m > h \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{11,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is immediately left of the head word.

$$\mathbf{f}_{11,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \text{ and } m = h - 1 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{12,a,b}(x_{1:n}, h, m, l)$ : The head word is $a$, the modifier is $b$, and the modifier is immediately right of the head word.

$$\mathbf{f}_{12,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } word(x_h) = a \text{ and } word(x_m) = b \text{ and } m = h + 1 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{13,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is to the left of the head word.

$$\mathbf{f}_{13,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \text{ and } m < h \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{14,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is to the right of the head word.

$$\mathbf{f}_{14,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \text{ and } m > h \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{15,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is immediately left of the head word.

$$\mathbf{f}_{15,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \text{ and } m = h - 1 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{16,a,b}(x_{1:n}, h, m, l)$ : The head word PoS is $a$, the modifier PoS is $b$, and the modifier is immediately right of the head word.

$$\mathbf{f}_{16,a,b}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = a \text{ and } pos(x_m) = b \text{ and } m = h + 1 \\ 0 & \text{otherwise} \end{cases}$$

(c) Tags between head and modifier

- $\mathbf{f}_{17,a}(x_{1:n}, h, m, l)$ : The PoS tag $a$ appears between the modifier and the head word.

$$\mathbf{f}_{17,a}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } \exists i : m < i < h \vee h < i < m : pos(x_i) = a \\ 0 & \text{otherwise} \end{cases}$$

(d) S-V-O variations

- $\mathbf{f}_{18}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its left, and it is the subject.

$$\mathbf{f}_{18}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = \text{V and } m < h \text{ and } l = \text{subj} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{19}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its right, and it is the subject.

$$\mathbf{f}_{19}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = \text{V and } m > h \text{ and } l = \text{subj} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{20}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its left, and it is the object.

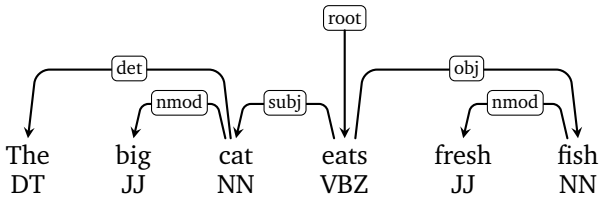
$$\mathbf{f}_{20}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = \text{V and } m < h \text{ and } l = \text{obj} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{f}_{21}(x_{1:n}, h, m, l)$ : The head is a verb, the modifier is to its right, and it is the object.

$$\mathbf{f}_{21}(x_{1:n}, h, m, l) = \begin{cases} 1 & \text{if } pos(x_h) = \text{V and } m > h \text{ and } l = \text{obj} \\ 0 & \text{otherwise} \end{cases}$$

2. We need to apply the patterns to each arc in the tree, and sum the binary features to get the feature vector $f(x, y)$ for the given pair.

   The given tree $y$ is:




   The features we get for each edge in $y$ are:

| edge | features |
|---|---|
| $(3, 1, \text{det})$ | $\mathbf{f}_{1,\text{NN,DT}}$; $\mathbf{f}_{2,\text{NN,DT,det}}$; $\mathbf{f}_{3,cat,the}$; $\mathbf{f}_{4,cat,the,\text{det}}$; $\mathbf{f}_5$; $\mathbf{f}_{9,cat,the}$; $\mathbf{f}_{13,\text{NN,DT}}$; $\mathbf{f}_{17,\text{JJ}}$ |
| $(3, 2, \text{nmod})$ | $\mathbf{f}_{1,\text{NN,JJ}}$; $\mathbf{f}_{2,\text{NN,JJ,nmod}}$; $\mathbf{f}_{3,cat,big}$; $\mathbf{f}_{4,cat,big,\text{nmod}}$; $\mathbf{f}_5$; $\mathbf{f}_7$; $\mathbf{f}_{9,cat,big}$; $\mathbf{f}_{11,cat,big}$; $\mathbf{f}_{13,\text{NN,JJ}}$; $\mathbf{f}_{15,\text{NN,JJ}}$ |
| $(4, 3, \text{subj})$ | $\mathbf{f}_{1,\text{VBZ,NN}}$; $\mathbf{f}_{2,\text{VBZ,NN,subj}}$; $\mathbf{f}_{3,eats,cat}$; $\mathbf{f}_{4,eats,cat,\text{subj}}$; $\mathbf{f}_5$; $\mathbf{f}_7$; $\mathbf{f}_{9,eats,cat}$; $\mathbf{f}_{11,eats,cat}$; $\mathbf{f}_{13,\text{VBZ,NN}}$; $\mathbf{f}_{15,\text{VBZ,NN}}$; $\mathbf{f}_{18}$ |
| $(0, 4, \text{root})$ | $\mathbf{f}_{1,*,\text{VBZ}}$; $\mathbf{f}_{2,*,\text{VBZ,root}}$; $\mathbf{f}_{3,*,eats}$; $\mathbf{f}_{4,*,eats,\text{root}}$ |
| $(6, 5, \text{nmod})$ | $\mathbf{f}_{1,\text{NN,JJ}}$; $\mathbf{f}_{2,\text{NN,JJ,nmod}}$; $\mathbf{f}_{3,fish,fresh}$; $\mathbf{f}_{4,fish,fresh,\text{nmod}}$; $\mathbf{f}_5$; $\mathbf{f}_7$; $\mathbf{f}_{9,fish,fresh}$; $\mathbf{f}_{11,fish,fresh}$; $\mathbf{f}_{13,\text{NN,JJ}}$; $\mathbf{f}_{15,\text{NN,JJ}}$ |
| $(4, 6, \text{obj})$ | $\mathbf{f}_{1,\text{VBZ,NN}}$; $\mathbf{f}_{2,\text{VBZ,NN,obj}}$; $\mathbf{f}_{3,eats,fish}$; $\mathbf{f}_{4,eats,fish,\text{obj}}$; $\mathbf{f}_6$; $\mathbf{f}_{10,eats,fish}$; $\mathbf{f}_{17,\text{JJ}}$; $\mathbf{f}_{21}$ |

   Thus, the complete $f(x, y)$ vector is:

| | |
|---|---|
| $\mathbf{f}_{1,\text{NN,DT}}$ | 1 |
| $\mathbf{f}_{1,\text{NN,JJ}}$ | 2 |
| $\mathbf{f}_{1,\text{VBZ,NN}}$ | 2 |
| $\mathbf{f}_{1,*,\text{VBZ}}$ | 1 |
| $\mathbf{f}_{2,\text{NN,DT,det}}$ | 1 |
| $\mathbf{f}_{2,\text{NN,JJ,nmod}}$ | 2 |
| $\mathbf{f}_{2,\text{VBZ,NN,subj}}$ | 1 |
| $\mathbf{f}_{2,*,\text{VBZ,root}}$ | 1 |
| $\mathbf{f}_{2,\text{VBZ,NN,obj}}$ | 1 |
| $\mathbf{f}_{3,cat,the}$ | 1 |
| $\mathbf{f}_{3,cat,big}$ | 1 |
| $\mathbf{f}_{3,eats,cat}$ | 1 |
| $\mathbf{f}_{3,*,eats}$ | 1 |
| $\mathbf{f}_{3,fish,fresh}$ | 1 |
| $\mathbf{f}_{3,eats,fish}$ | 1 |

| | |
|---|---|
| $\mathbf{f}_{4,cat,the,\text{det}}$ | 1 |
| $\mathbf{f}_{4,cat,big,\text{nmod}}$ | 1 |
| $\mathbf{f}_{4,eats,cat,\text{subj}}$ | 1 |
| $\mathbf{f}_{4,*,eats,\text{root}}$ | 1 |
| $\mathbf{f}_{4,fish,fresh,\text{nmod}}$ | 1 |
| $\mathbf{f}_{4,eats,fish,\text{obj}}$ | 1 |
| $\mathbf{f}_5$ | 4 |
| $\mathbf{f}_6$ | 1 |
| $\mathbf{f}_7$ | 3 |
| $\mathbf{f}_{9,cat,the}$ | 1 |
| $\mathbf{f}_{9,cat,big}$ | 1 |
| $\mathbf{f}_{9,eats,cat}$ | 1 |
| $\mathbf{f}_{9,fish,fresh}$ | 1 |
| $\mathbf{f}_{10,eats,fish}$ | 1 |

| | |
|---|---|
| $\mathbf{f}_{11,cat,big}$ | 1 |
| $\mathbf{f}_{11,eats,cat}$ | 1 |
| $\mathbf{f}_{11,fish,fresh}$ | 1 |
| $\mathbf{f}_{13,\text{NN,DT}}$ | 1 |
| $\mathbf{f}_{13,\text{NN,JJ}}$ | 2 |
| $\mathbf{f}_{13,\text{VBZ,NN}}$ | 1 |
| $\mathbf{f}_{14,\text{VBZ,NN}}$ | 1 |
| $\mathbf{f}_{15,\text{NN,JJ}}$ | 2 |
| $\mathbf{f}_{15,\text{VBZ,NN}}$ | 1 |
| $\mathbf{f}_{17,\text{JJ}}$ | 2 |
| $\mathbf{f}_{18}$ | 1 |
| $\mathbf{f}_{21}$ | 1 |

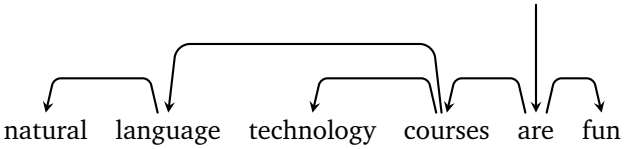
## Exercise 6.

Given the sentence *natural language technology courses are fun*,

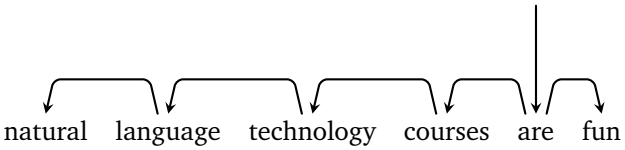
1. Draw unlabeled dependency trees for the following interpretations
   (a) technology courses about natural language are fun
   (b) courses about technology on natural language are fun
   (c) natural courses about language technology are fun
   (d) courses about natural technology for language are fun
2. Emulate the behaviour of a transition dependency parser using an arc-standard model (i.e. with operations *shift*, *left-arc*, and *right-arc* between the two topmost stack elements). List the intermediate stack/buffer contents and the selected action at each step needed to obtain the tree for each of the interpretations above.

## SOLUTION

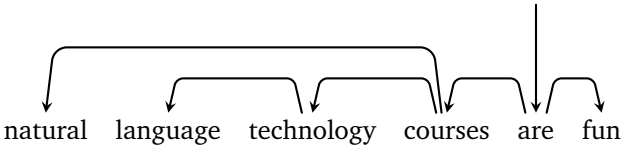
1. (a) technology courses about natural language are fun



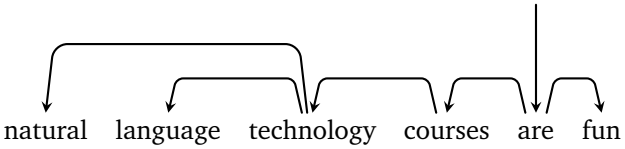

(b) courses about technology on natural language are fun




(c) natural courses about language technology are fun




(d) courses about natural technology for language are fun



2. Transition sequences to build each tree

(a) technology courses about natural language are fun

| Stack | Buffer | Transition | Edges |
|---|---|---|---|
| * | natural language technology courses are fun | sh | {} |
| * natural | language technology courses are fun | sh | {} |
| * natural language | technology courses are fun | l-arc | {} |
| * language | technology courses are fun | sh | {(2,1)} |
| * language technology | courses are fun | sh | {(2,1)} |
| * language technology courses | are fun | l-arc | {(2,1)} |
| * language courses | are fun | l-arc | {(2,1),(4,3)} |
| * courses | are fun | sh | {(2,1),(4,3),(4,2)} |
| * courses are | fun | l-arc | {(2,1),(4,3),(4,2)} |
| * are | fun | sh | {(2,1),(4,3),(4,2),(5,4)} |
| * are fun | | r-arc | {(2,1),(4,3),(4,2),(5,4)} |
| * are | | r-arc | {(2,1),(4,3),(4,2),(5,4),(5,6)} |
| * | | stop | {(2,1),(4,3),(4,2),(5,4),(5,6),(0,5)} |

(b) courses about technology on natural language are fun

| Stack | Buffer | Transition | Edges |
|---|---|---|---|
| * | natural language technology courses are fun | sh | {} |
| * natural | language technology courses are fun | sh | {} |
| * natural language | technology courses are fun | l-arc | {} |
| * language | technology courses are fun | sh | {(2,1)} |
| * language technology | courses are fun | l-arc | {(2,1)} |
| * technology | courses are fun | sh | {(2,1),(3,2)} |
| * technology courses | are fun | l-arc | {(2,1),(3,2)} |
| * courses | are fun | sh | {(2,1),(3,2),(4,3)} |
| * courses are | fun | l-arc | {(2,1),(3,2),(4,3)} |
| * are | fun | sh | {(2,1),(3,2),(4,3),(5,4)} |
| * are fun | | r-arc | {(2,1),(3,2),(4,3),(5,4)} |
| * are | | r-arc | {(2,1),(3,2),(4,3),(5,4),(5,6)} |
| * | | stop | {(2,1),(3,2),(4,3),(5,4),(5,6),(0,5)} |

(c) natural courses about language technology are fun

| Stack | Buffer | Transition | Edges |
|---|---|---|---|
| * | natural language technology courses are fun | sh | {} |
| * natural | language technology courses are fun | sh | {} |
| * natural language | technology courses are fun | sh | {} |
| * natural language technology | courses are fun | l-arc | {} |
| * natural technology | courses are fun | sh | {(3,2)} |
| * natural technology courses | are fun | l-arc | {(3,2)} |
| * natural courses | are fun | l-arc | {(3,2),(4,3)} |
| * courses | are fun | sh | {(3,2),(4,3),(4,1)} |
| * courses are | fun | l-arc | {(3,2),(4,3),(4,1),(5,4)} |
| * are | fun | sh | {(3,2),(4,3),(4,1),(5,4)} |
| * are fun | | r-arc | {(3,2),(4,3),(4,1),(5,4)} |
| * are | | r-arc | {(3,2),(4,3),(4,1),(5,4),(5,6)} |
| * | | stop | {(3,2),(4,3),(4,1),(5,4),(5,6),(0,5)} |

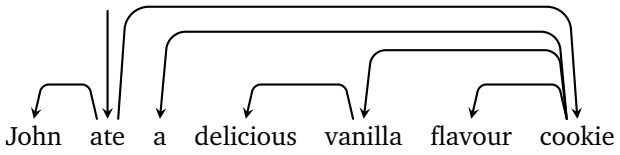
(d) courses about natural technology for language are fun

| Stack | Buffer | Transition | Edges |
|---|---|---|---|
| * | natural language technology courses are fun | sh | {} |
| * natural | language technology courses are fun | sh | {} |
| * natural language | technology courses are fun | sh | {} |
| * natural language technology | courses are fun | l-arc | {} |
| * natural technology | courses are fun | l-arc | {(3,2)} |
| * technology | courses are fun | sh | {(3,2),(3,1)} |
| * technology courses | are fun | l-arc | {(3,2),(3,1)} |
| * courses | are fun | sh | {(3,2),(3,1),(4,3)} |
| * courses are | fun | l-arc | {(3,2),(3,1),(4,3),(5,4)} |
| * are | fun | sh | {(3,2),(3,1),(4,3),(5,4)} |
| * are fun | | r-arc | {(3,2),(3,1),(4,3),(5,4)} |
| * are | | r-arc | {(3,2),(3,1),(4,3),(5,4),(5,6)} |
| * | | stop | {(3,2),(3,1),(4,3),(5,4),(5,6),(0,5)} |

## Exercise 7. Parsing

Given the sentence *John ate a delicious vanilla flavour cookie*,

1. Draw unlabeled dependency trees for the following interpretations
   (a) John ate a cookie with flavour of delicious vanilla
   (b) John ate a delicious cookie with vanilla flavour
   (c) John ate a delicious and flavoured cookie made of vanilla
   (d) John ate a cookie with a delicious flavour of vanilla
2. Given the tree

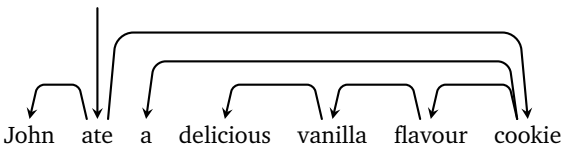



   (a) Explain the interpretation encoded by this tree avoiding any ambiguities.
   (b) Emulate the behaviour that would result in this tree for a transition dependency parser using an arc-standard model (i.e. with operations *shift*, *left-arc*, and *right-arc* between the two topmost stack elements). List the intermediate stack/buffer contents and the required action at each step to obtain the final tree.

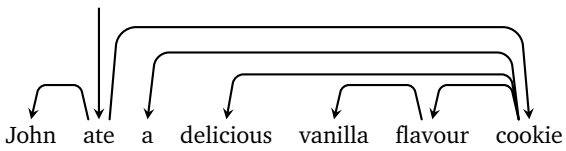
## SOLUTION

1.

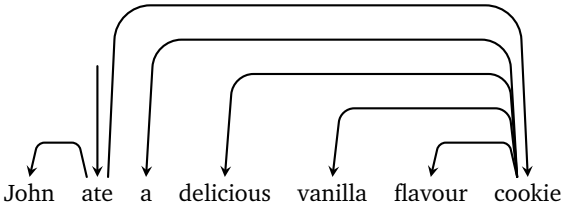
(a)



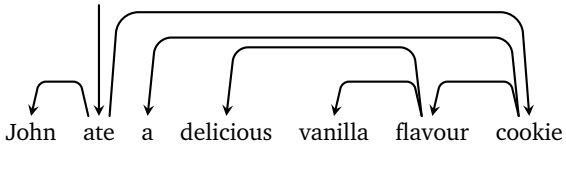

(b)




(c)




(d)




2.

   (a) The tree represents the interpretation where John ate a flavoured cookie made of delicious vanilla.
   (b) The behaviour of an arc-based transition parser to obtain this interpretation would be the following:

| Stack | Buffer | Transition | Edges |
|---|---|---|---|
| * | John ate a delicious vanilla flavour cookie | sh | {} |
| * John | ate a delicious vanilla flavour cookie | sh | {} |
| * John ate | a delicious vanilla flavour cookie | l-arc | {(2,1)} |
| * ate | a delicious vanilla flavour cookie | sh | {(2,1)} |
| * ate a | delicious vanilla flavour cookie | sh | {(2,1)} |
| *ate a delicious | vanilla flavour cookie | sh | {(2,1)} |
| * ate a delicious vanilla | flavour cookie | l-arc | {(2,1),(5,4)} |
| * ate a vanilla | flavour cookie | sh | {(2,1),(5,4)} |
| * ate a vanilla flavour | cookie | sh | {(2,1),(5,4)} |
| * ate a vanilla flavour cookie | | l-arc | {(2,1),(5,4),(7,6)} |
| * ate a vanilla cookie | | l-arc | {(2,1),(5,4),(7,6),(7,5)} |
| * ate a cookie | | l-arc | {(2,1),(5,4),(7,6),(7,5),(7,3)} |
| * ate cookie | | r-arc | {(2,1),(5,4),(7,6),(7,5),(7,3),(2,7)} |
| * ate | | r-arc | {(2,1),(5,4),(7,6),(7,5),(7,3),(2,7),(0,2)} |
| * | | stop | {(2,1),(5,4),(7,6),(7,5),(7,3),(2,7),(0,2)} |

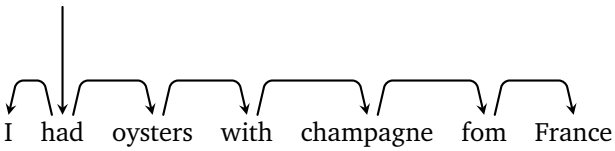
## Exercise 8.

Given the sentence *I had oysters with champagne from France*.

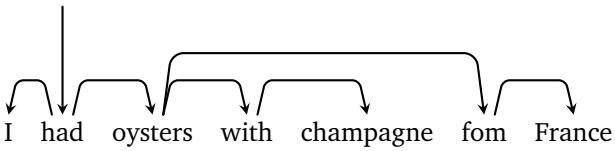
1. Draw unlabeled dependency trees for the following interpretations:
   (a) I had oysters which had champagne on them. The champagne was from France.
   (b) I had oysters which had champagne on them. The oysters were from France.
   (c) I had oysters while having also champagne. The champagne was from France.
   (d) I had oysters while having also champagne. The oysters were from France.
2. Is any of the obtained trees non-projective? Justify your answer.

## SOLUTION

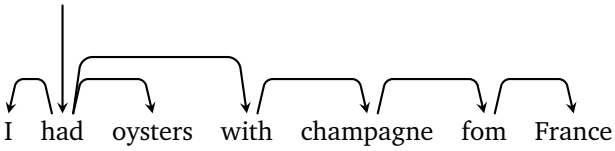
1. Draw unlabeled dependency trees for the following interpretations:
   (a) I had oysters which had champagne on them. The champagne was from France.



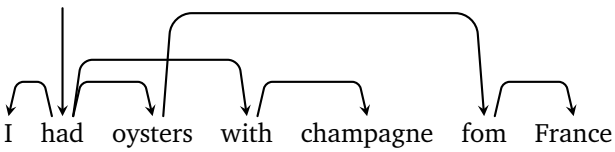

   (b) I had oysters which had champagne on them. The oysters were from France.




   (c) I had oysters while having also champagne. The champagne was from France.




   (d) I had oysters while having also champagne. The oysters were from France.




2. Structure (d) is non-projective, since there are crossing arcs.

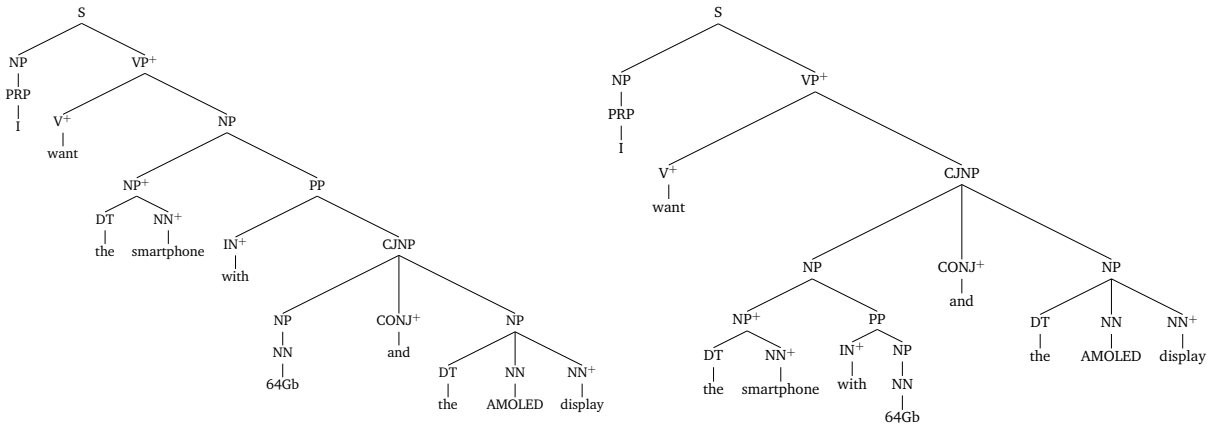
## Exercise 9.

A Papazom.com user wrote the sentence:

*I want the smartphone with 64Gb and the AMOLED display*

We used the following PCFG grammar (where the $^+$ superscript indicates the head of each rule), and obtained the two possible parse trees below.

| | | | | | |
|---|---|---|---|---|---|
| $r_1$ S $\rightarrow$ NP VP$^+$ | 1.0 | $r_7$ NP $\rightarrow$ PRP | 0.1 | $r_{12}$ PRP $\rightarrow$ *I* | 1.0 |
| $r_2$ VP $\rightarrow$ V$^+$ NP | 0.7 | $r_8$ NP $\rightarrow$ NN | 0.2 | $r_{13}$ V $\rightarrow$ *want* | 1.0 |
| $r_3$ VP $\rightarrow$ V$^+$ CJNP | 0.3 | $r_9$ NP $\rightarrow$ NP$^+$ PP | 0.2 | $r_{14}$ CONJ $\rightarrow$ *and* | 1.0 |
| $r_4$ CJNP $\rightarrow$ NP CONJ$^+$ NP | 1.0 | $r_{10}$ NP $\rightarrow$ DT NN$^+$ | 0.3 | $r_{15}$ DT $\rightarrow$ *the* | 1.0 |
| $r_5$ PP $\rightarrow$ IN$^+$ NP | 0.6 | $r_{11}$ NP $\rightarrow$ DT NN NN$^+$ | 0.2 | $r_{16}$ IN $\rightarrow$ *with* | 1.0 |
| $r_6$ PP $\rightarrow$ IN$^+$ CJNP | 0.4 | | | $r_{17}$ NN $\rightarrow$ *display* | 0.3 |
| | | | | $r_{18}$ NN $\rightarrow$ *smartphone* | 0.4 |
| | | | | $r_{19}$ NN $\rightarrow$ 64*Gb* | 0.2 |
| | | | | $r_{20}$ NN $\rightarrow$ *AMOLED* | 0.1 |




1. Which parse tree has higher probability according to the PCFG? Reason your answer.
2. Convert both parse trees to dependency trees.
3. Describe in an unambiguous form what is the meaning of the interpretation represented by each tree.

## SOLUTION

1. The probability of a parse tree is computed as $P(t) = \prod_{r \in t} q(r)$, that is, the product of the probabilities of the rules used to build it.

   Both trees use *each rule* in the grammar exactly *once*, with only two exceptions:

   - The first tree does not use $r_3$ nor $r_5$.
   - The second tree does not use $r_2$ nor $r_6$.

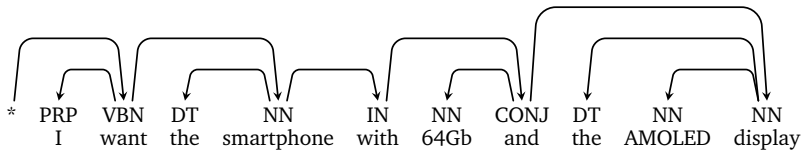
   Let $Q$ be the product of the probabilities of the rules common to both trees,
   i.e. $Q = \prod_{r \in R - \{r_2, r_3, r_5, r_6\}} P(r)$.

   Then, the first tree will have probability $P_1 = Q \times P(r_2) \times P(r_6) = Q \times 0.7 \times 0.4 = Q \times 0.28$, while the second tree will have probability $P_2 = Q \times P(r_3) \times P(r_5) = Q \times 0.3 \times 0.6 = Q \times 0.18$.

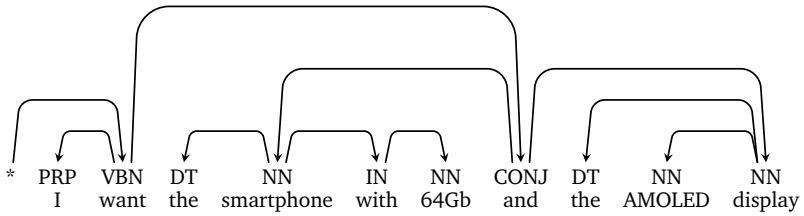
   Since $Q \times 0.28 > Q \times 0.18$, the first tree has higher probability according to this PCFG.

2. Convert both parse trees to dependency trees.

   First tree:




   Second tree:




3. Describe in an unambiguous form what is the meaning of the interpretation represented by each tree.

   First tree has the interpretation that the user wants a smartphone that has two features: 64Gb and an AMOLED display

   The second tree interpretation is that the user wants two products: One smartphone with 64Gb, and also one AMOLED display.