LING/C SC 581:
Advanced Computational Linguistics

Lecture 21

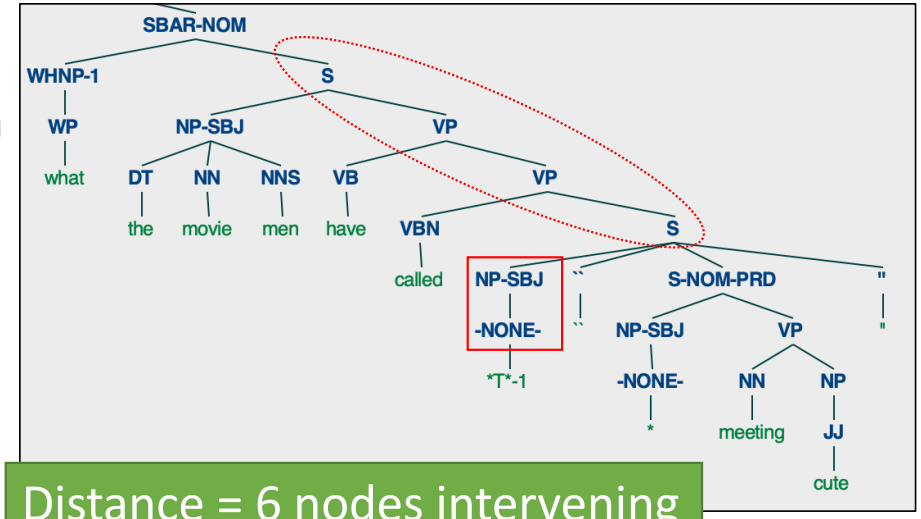
Today's Topic

- Homework 9 Review contd.
- *live* programming
- Homework 10
  - involve computing relations built on c-command
  - **a bit simplified:** some parts can be done manually, no coding needed

Homework 9: Question 5 Review

Modify your code to compute the distance between the antecedent and its trace.

- Find the tree with the biggest distance in the ptb corpus.
- **Hint:**
  - modify the function `dom` to count depth
  - Note: `dom` is recursive
  - use an extra parameter in the function call to increment the depth each time around



Homework 9: Question 5 Review

- Idea: increment the count with recursion

```
6 def dom(x):
7     yield x
8     if not isinstance(x, str):
9         for y in x:
10            yield from dom(y)
```

- Call:
  - `dom(x, 0)`
- Definition:
  - `dom(x, n)`
- Recursive call:
  - `dom(x, n+1)`
- `Yield` (normally `return`):
  - `x, n`
- Call return:
  - `w, i = dom(z, 0)`

Aside: Max in Python

- Basics:
  - `mx = 0` and somewhere in your code
  - `if depth > mx:`
  - `    mx = depth`

Homework 9: Question 5 Review

- How to get the tree (and its number)?
  - iterate over `ptb.parsed_sents()`
  - use built-in `enumerate()`
  - https://docs.python.org/3/library/functions.html#enumerate

```
>>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']
>>> list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
>>> list(enumerate(seasons, start=1))
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
```

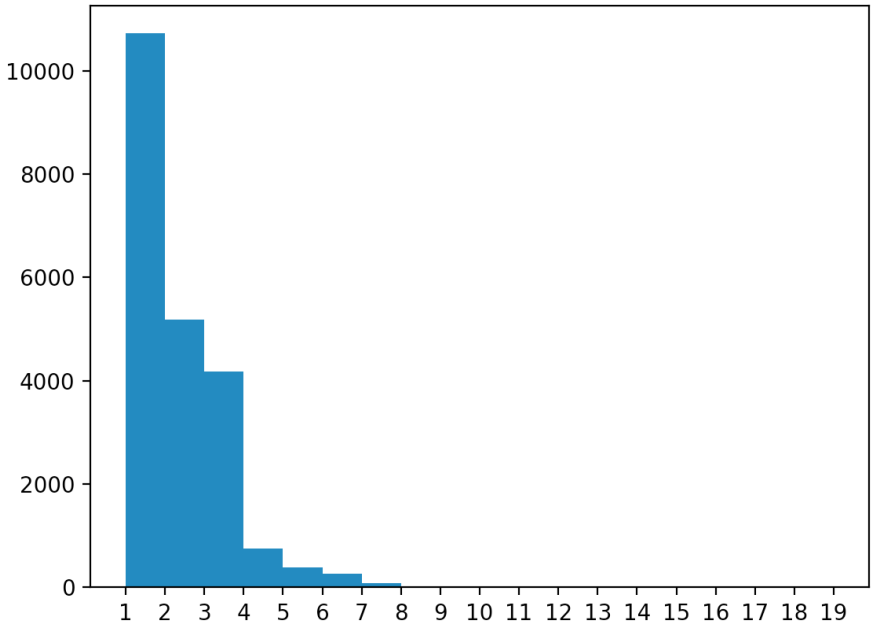
  - pass the tree number down to `cc()`
    - i.e. `cc(x, num)`
  - or access num as a global from inside `cc()`

Homework 9: Question 6 Review

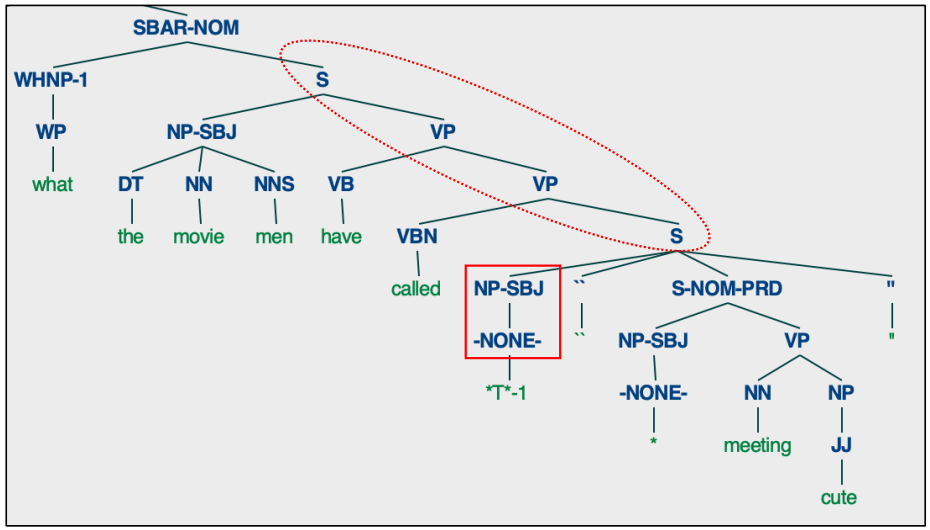
Extra Credit

- Modify your code to plot the histogram for the WH-antecedent to trace depth over the entire ptb corpus
- **Hint**:
  - `import matplotlib.pyplot as plt`
  - `plt.hist(dist, bins=range(1,mx))`
  - `plt.xticks(range(1,mx))`
  - `plt.show()`

Distance between trace and antecedent



- counting internal nodes only
- count – 2 used to exclude POS tag –NONE– and parent label



Aside: List of values in Python

- Basics:
  - `dist = []` and somewhere in your code `dist.append(value)`
- Then:
  - `plt.hist(dist, bins=range(1,mx))`

Homework 10

IMPORTANT:

- you may find it helpful to review the slides/Panopto from Tuesday …

- Consider subcorpus:
  - `ptb.parsed_sents()` index range `[70000:73451]`
  - `len(ptb.parsed_sents())`

    73451
  - `ptb.parsed_sents()[`*`index`*`].draw()`
  - `ptb.parsed_sents()[`*`index`*`].pretty_print()`
- Start with supplied Python code `ccommand3f.py`

Homework 10

```
python -i ccommand3f.py
>>>
```

ccommand3f.py

```
11 def dom(x, path):¶
12     if path is None:¶
13         path = list()¶
14     yield x, path¶
15     if not isinstance(x, str):¶
16         path.append(x.label())¶
17         for y in x:¶
18             yield from dom(y, path.copy())¶
```

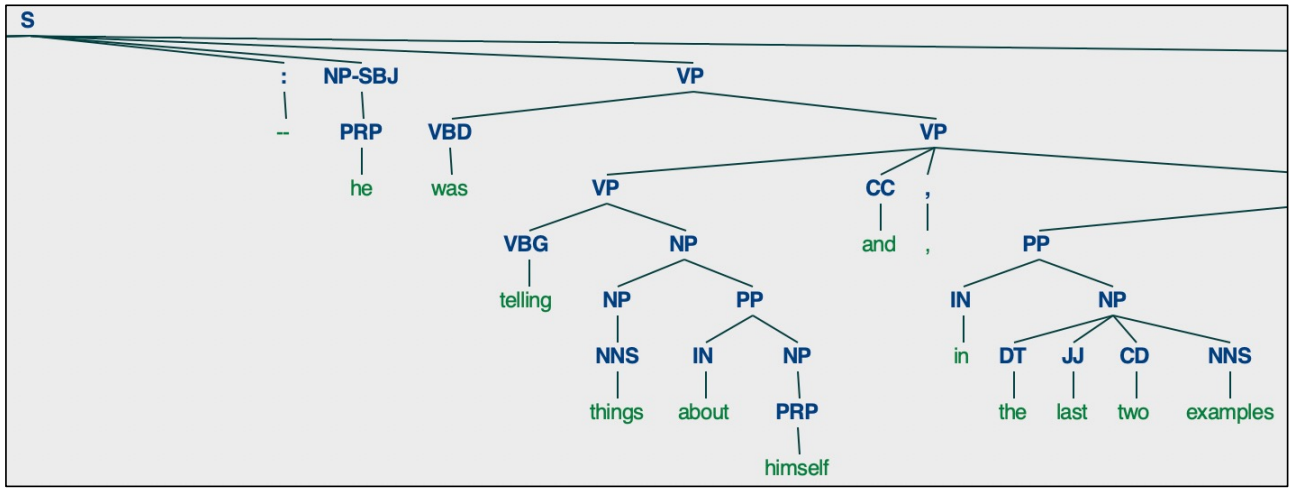
```
20 def cc(x):¶
21     if not isinstance(x, str):¶
22         if len(x) > 1:¶
23             for y,z in permutations(x, 2):¶
24                 m1 = re.search(yregex, y.label())¶
25                 if m1:¶
26                     for w, path in dom(z, None):¶
27                         if isinstance(w, str):¶
28                             m2 = re.search(wregex, w)¶
29                         else:¶
30                             m2 = re.search(wregex, w.label())¶
31                         if m2:¶
32                             print(y, 'c-commands', w, 'path', path)¶
33             for u in x:¶
34                 cc(u)¶
35         else:¶
36             cc(x[0])¶
```

Homework 10: Question 1

- Define appropriate global variables yregex and wregex to find candidate c-commanding NPs for anaphors ending in *–self*
- Search `ptb.parsed_sents()[70000:73451]`
- Show your code and work.
- How many examples of NP c-commanding anaphors are there?

Homework 10: Question 2

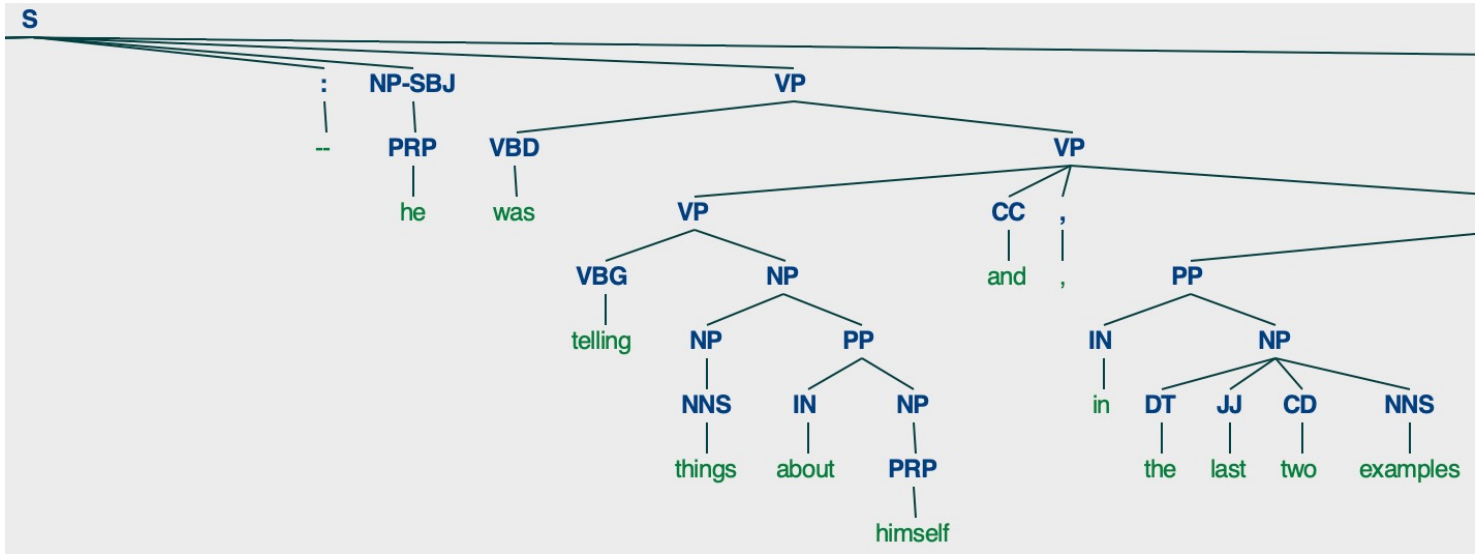
- Search `ptb.parsed_sents()[70000:73451]` again
- How many anaphors have more than one (candidate) c-commanding NP in the same tree?
- Example:



Homework 10: Question 2

- **Example**: note also the path computed by dom()

1. `(NP-SBJ (PRP he)) c-commands himself path ['VP', 'VP', 'VP', 'NP', 'PP', 'NP', 'PRP']`
2. `(NP (NNS things)) c-commands himself path ['PP', 'NP', 'PRP']`



Homework 10: Question 2

- You can count this *manually* from the output.
  - (*I've made the subcorpus small enough to do this.*)
- But you still need to distinguish outputs from different trees vs. the same tree.
- Two possible solutions:
  - Modify cc() to make a mark when you process a new tree, e.g. using print().
  - Use enumerate() when iterating over the corpus and access the tree number from within cc() when outputting the c-command relation.
  - *other approaches also possible*…

Homework 10: Question 3

- Suppose in the case of multiple candidate NPs, we adopt the rule:
  - the closest NP is the antecedent of the anaphor
- How well does this work in our subcorpus?
- **No need to implement.**
- Give both positive and negative examples.

Homework 10: Question 4

- How might you change/modify the closest NP rule to improve its accuracy?
- Describe its effect with examples.
- **No need to implement.**