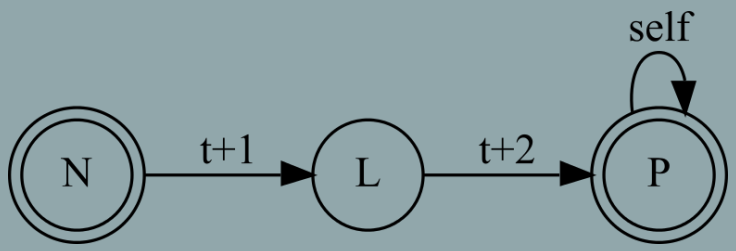
LING-362

Introduction to **N**atural **L**anguage **P**rocessing

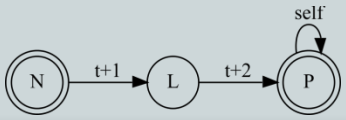
Syntactic parsing III



Review - CKY

- Tokens at the top of the table

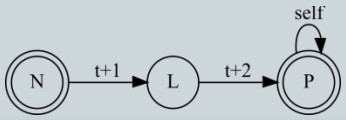
The	boys	shop	daily



Dynamic programming – CKY

- Check bottom of column for possible categories for each token

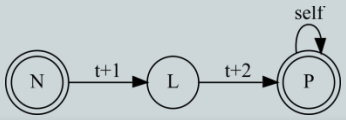
The	boys	shop	daily
DT			



Dynamic programming – CKY

- Check intersections in higher rows:
  - Is this a possible segmentation?
  - Only binary branching grammars allowed (Chomsky Normal Form – **CNF**)

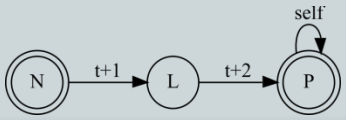
The	boys	shop	daily
DT			
	NNS		



Dynamic programming – CKY

- Unambiguous – no problem

The	boys	shop	daily
DT	NP		
	NNS		

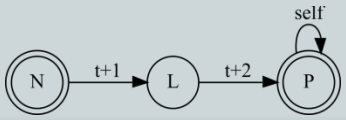


Dynamic programming – CKY

- For ambiguous cases, we must keep track of what goes with what

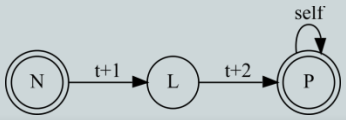
Derivation does not terminate!

The	boys	shop	daily
DT	NP		S | S
	NNS	NP	
		VBP | NN	VP
			RB | NN



Choosing the right parse

- All an algorithm like CKY does for us so far is check for **possible** positions to **split into phrases**
  - Useful in many contexts (CKY can be applied to Chinese word segmentation! Qian & Liu 2012)
  - Genome mapping (see Poptsova 2014)
  - …
- How can we decide which parse is right?

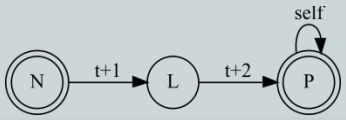


From CFGs to PCFGs

- We can amend our definition to include probabilities – a **Probabilistic CFG**:

  G ≡

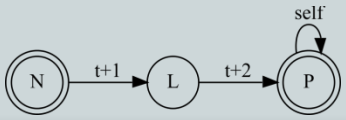
N	Set of **non-terminal** symbolds
Σ	Set of **terminal** symbols (not in N!)
R	Set of rules of the form A → β [p] where A ∈ N, β ∈ (Σ∪N)* and **p** is the probability P(β|A)
S	The designated start symbol



From CFGs to PCFGs

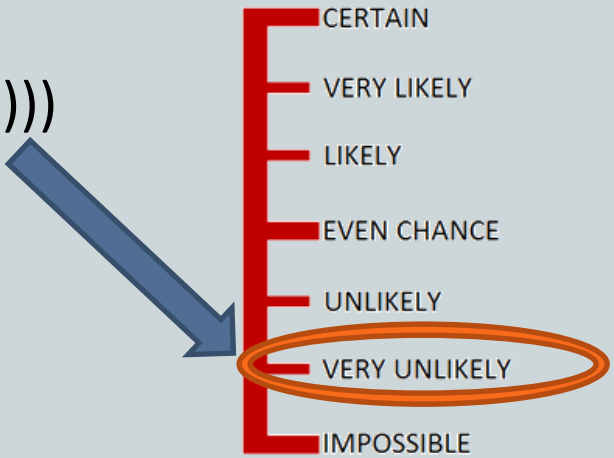
- Probabilities?
  - We can treat the proportion of each decomposition of each category as its probability
    - NP > DT NN: 35%
    - NP > NNS: 20%
    - NP > PRP: 20%
    - NP > DT JJ NN: 10%
    - …

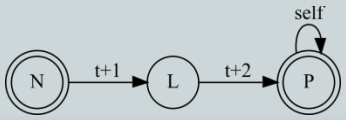
    100%



This sounds like a great idea!

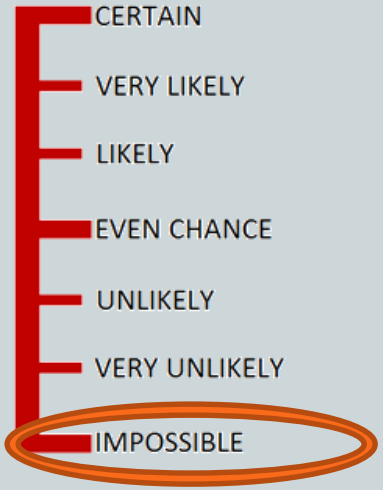
- Even if we have lots of spurious generations in our data, the parser will **never** choose them!
  - Sure, these constituents are licensed by the grammar:
    - (ADVP (NP (NNS finishes))))
    - (NP (DT the) (NP (NNS finishes)))
  - But they are very unlikely
  - This is good news, right?



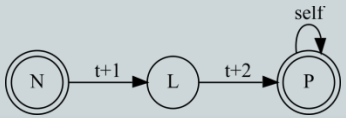


..parser will **never** choose them?

- The problem with ‘very unlikely’ in CFG is that it effectively means **never**
- Consider legitimate ambiguities:
  - PP attachment:
    - I [saw [the man with the telescope]]
    - I [saw [the man] [with the telescope]]
  - PCFG level view:
    - VP > V NP 65%
    - VP > V NP PP 35%

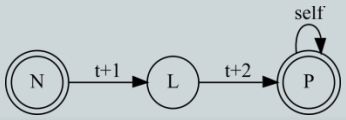


- Incapable of **ever** getting high attachment! ☹



Structural and lexical dependencies

- Which rules we apply depends on:
  - Lexical information (some verbs often have a high attached instrument and some prepositions mark these, e.g. ***break X with Y***)
  - Non-terminal, structural context: pronoun NPs are much more likely as subjects than objects
- How can we get these into our grammar



Band-aid 1

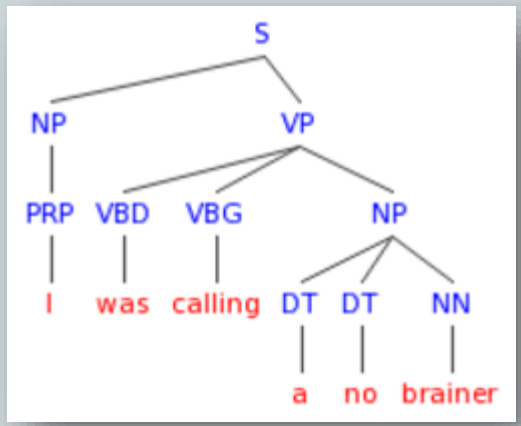
- We can annotate each internal node with its parent: (Johnson 1998)

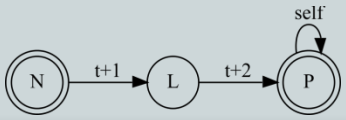
  (S

  (NP (PRP I))

  (VP (VBD was) (VBG calling)

  (NP (DT a) (DT no) (NN brainer))))

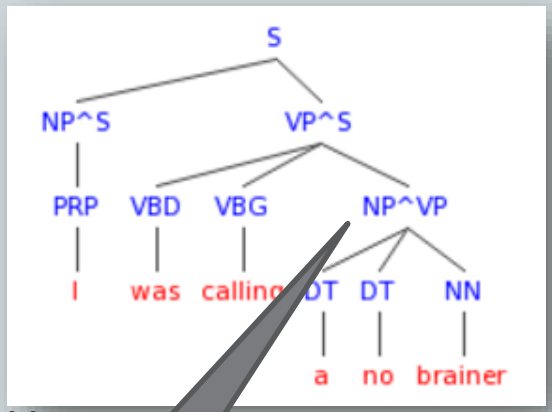




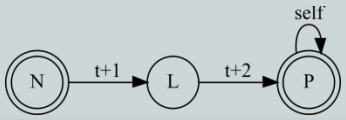
Band-aid 1

- We can annotate each internal node with its parent: (Johnson 1998)

```
(S
  (NP^S (PRP I))
  (VP^S (VBD was) (VBG calling)
      (NP^VP (DT a) (DT no) (NN brainer))))
```



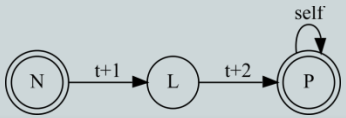
Parent Annotation



Pros and cons

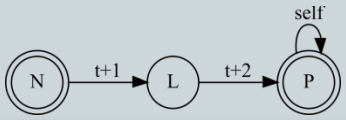
- Adding parent information effectively causes **label splitting**
  - Leads to data sparseness: less examples of each label
  - Some splits are useful, others are harmful – how can we tell?

- Initial attempts at choosing the right cases to split required a lot of manual work
- But paid off for English: 72% -> 86% accuracy (Klein & Manning 2003)



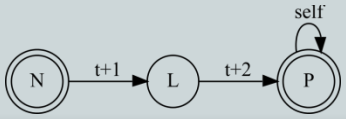
Why choose ourselves?

- Petrov et al. (2006) devised the **split and merge** algorithm
  - Automatically testing splitting of categories (88.4%)
  - Automatically merging categories (89.5%)
  - Best result with smoothing (90.2%)
  - State of the art into the 2010s, when neural networks and distributional semantics were added



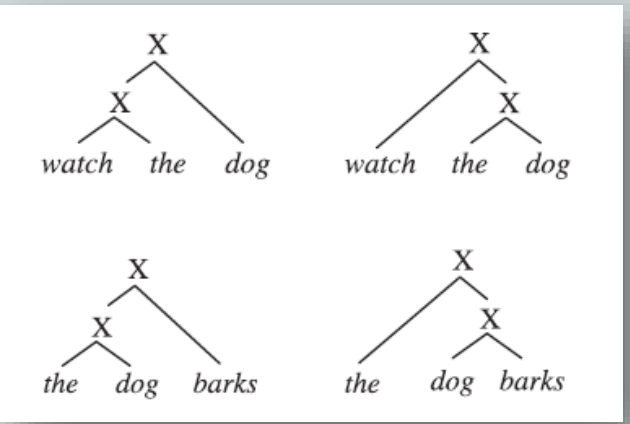
Excursus: cognitive implications

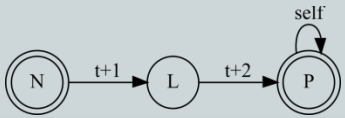
- It is very clear that we know more than CFGs
  - Lexicalized effects in processing, garden path sentences
  - Humans not bogged down by massive ambiguity options, unless lexical items collude with syntax
    - ***I saw the man with the telescope***
    - ***The cop chased the criminal with the fast car***



Excursus: cognitive implications

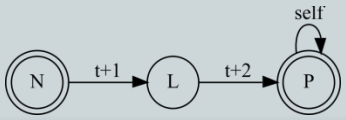
- What tree depths are we tracking?
  - According to Bod (2009), allowing a language acquisition simulation to condition on **depth 4** trees is optimal
  - Learning based on PTB and CHILDES (MacWhinney 2000)
  - Computers reproduce children's errors! ☺





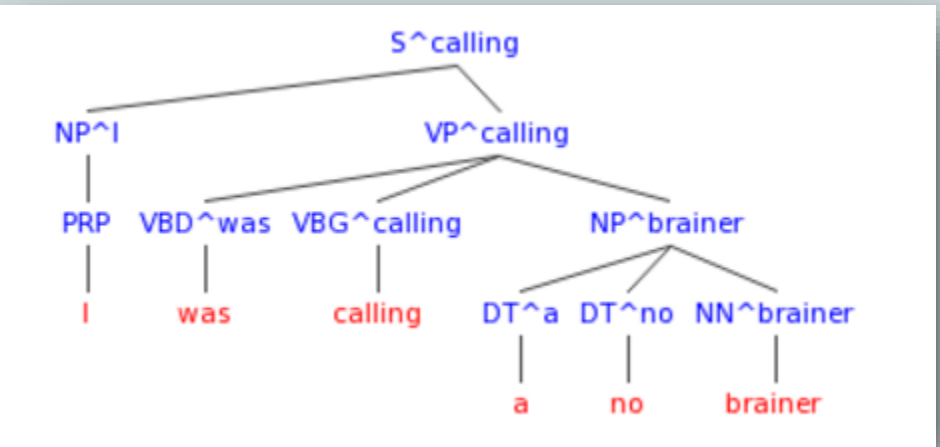
Band-aid 2

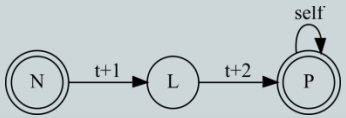
- Even label splitting doesn't help with **lexical** probabilities
  - Does is matter for PP attachment that the preposition is ***with***?
  - Does it matter whether the verb is ***chased*** or ***saw***?



Lexicalized grammars

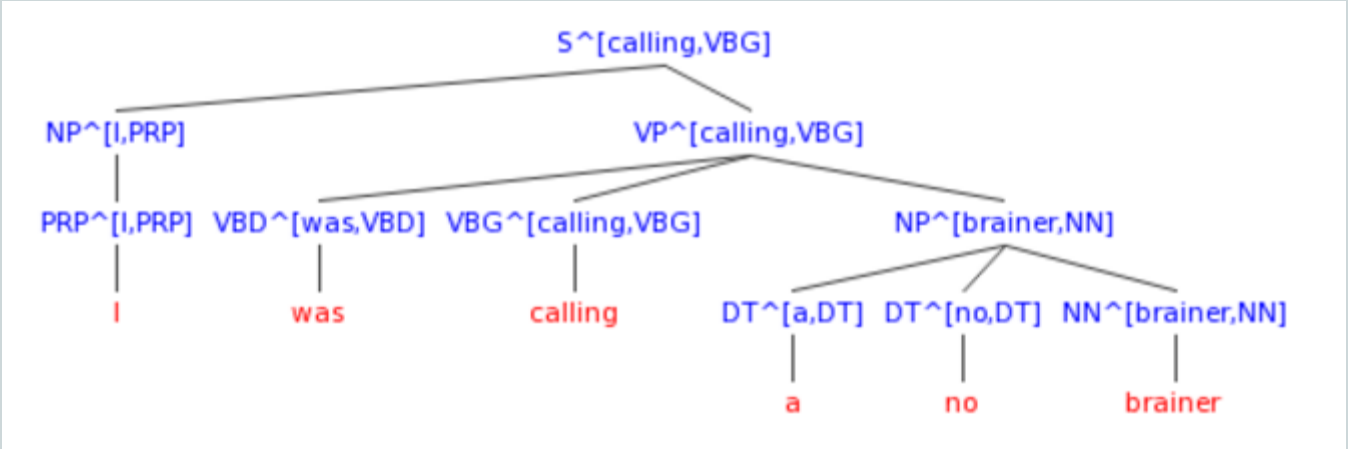
- An easy way to bring in lexical information is to annotate each node with its lexical head **(head percolation)**

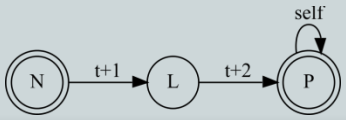




Lexicalized grammars

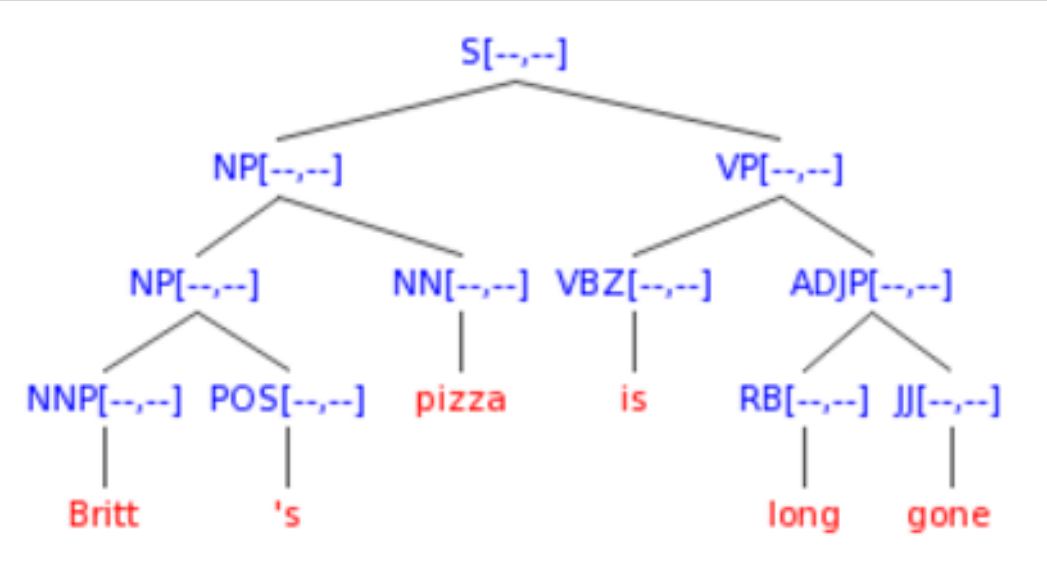
- In practice, most lexicalized parsers work on percolated head+POS annotation
- State of the art choice when data is limited/no large word embeddings available

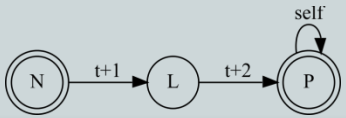




Quick exercise

- What needs to percolate where?

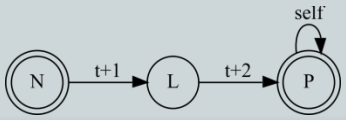




Data sparseness

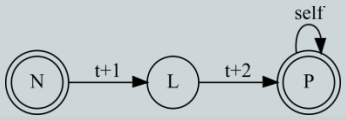
- A major problem with the lexicalized approach is sparseness
- We hardly get probabilities for rules like this:

VP[calling,VBG] > VBD[was,VBD] VBG[calling,VBG] NP[brainer,NN]



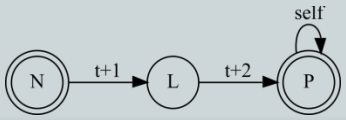
Generation probabilities

- Instead of considering the full rule:
  - VP[calling,VBG] > VBD[was,VBD] VBG[calling,VBG] NP[brainer,NN]
- We can pretend that the head gets generated first:
  - VP[calling,VBG] > VBG[calling,VBG]
- This part is still trivial but…



Generation probabilities

- Imagine we now consider each additional RHS component separately:
  - What is the chance that VBG[calling,VBG] has NP[brainer,NN] to the right?
  - P(NP[brainer,NN]|VBG[calling,VBG])
  - Maybe also not so common – but more so than an entire rule

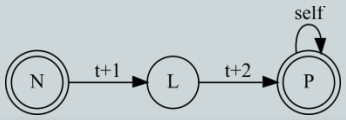


Generation probabilities

- We repeat this for all possible RHS members to the right of the head
- And do the same on the left

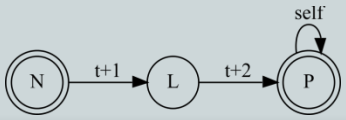
VP[calling,VBG] > VBG[calling,VBG]

VBD[was,VBD] ?← VBG[calling,VBG] →? NP[brainer,NN]



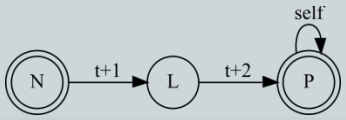
Generation probabilities

- What if there are two options?
  - calling > was calling $NP_{brainer}$
  - calling > was calling #
- We need to consider 'end of the line' as an item for probability estimation
  - Add 'fake' RHS member "STOP"
  - Estimate probability of stopping generation



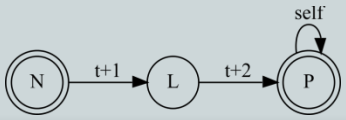
Implementation: Collins Parser

- The Collins parser implements this approach:
  - Estimate probability for lexicalized decomposition rules
  - Separately calculate probability of adding potential RHS constituent until STOP symbol is generated



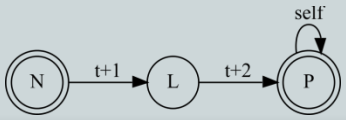
In Python

- There are many parsers out there
- Many developers use the **Stanford Parser**
  - Written in Java
  - A Python ‘wrapper’ exists
  - But you can really run any Java tool from your code!



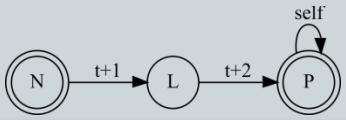
Calling the command line

```
import subprocess

command_params = ["java", "-mx2048m", "-cp" ,'"*;"',
"edu.stanford.nlp.parser.lexparser.LexicalizedParser" ,...]

proc = subprocess.Popen(command_params)
(stdout, stderr) = proc.communicate()

# your parse is now in stdout
```



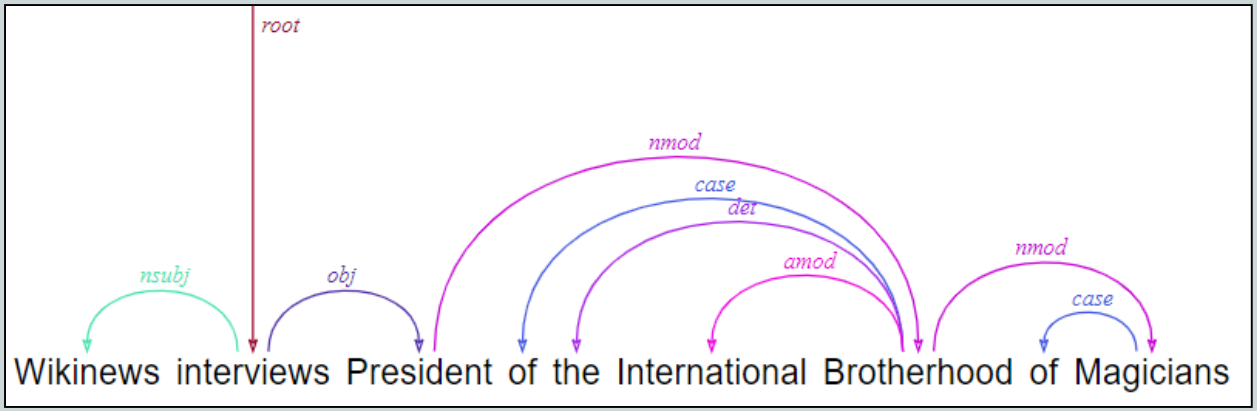
In Python

- Another option is spacy
  - pip install spacy
  - python -m spacy.en.download (takes long!)
- See https://spacy.io/ for a tutorial!
- For latest SOTA constituent parsers (not user friendly):
  - http://nlpprogress.com/english/constituency_parsing.html



In Python

- And many applications use **dependency parses** either instead of or in tandem with constituents



- Not covered in this course -> see LING-367 and more advanced classes