CS388: Natural Language Processing

Lecture 16: Syntax I

Greg Durrett
TEXAS
The University of Texas at Austin

Some slides adapted from Dan Klein, UC Berkeley

Administrivia

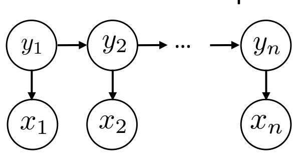
- Project 3 back soon
- FP check-ins due April 4

Recap: POS Tagging

- Layer of shallow syntactic analysis

NN NNS VBZ NNS	**VBP**	**NN**
Teacher strikes idle kids	I **record** the video	I listen to the **record**

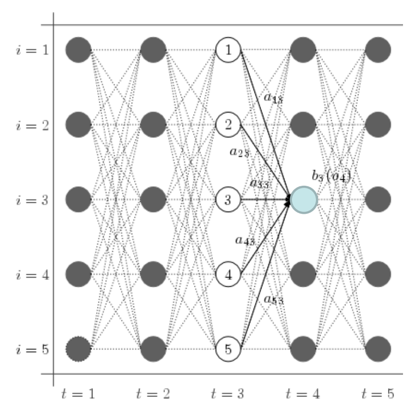
- One way to model it: Hidden Markov Models, generative models of P(**y**, **x**) from which we compute the posterior P(**y** | **x**) (+ use Viterbi to max)



$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} P(y_i | y_{i-1}) \prod_{i=1}^{n} P(x_i | y_i)$$

- Can also use conditional random fields (discriminative) or even **neural** CRFs — better for tasks like named entity recognition

Recap: Viterbi



- Best (partial) score for a sequence ending in state *s*

$$\boxed{\text{score}_1(s) = P(s)P(x_1|s)}$$

$$\boxed{\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})}$$

- Dynamic program allows us to efficiently compute the max-scoring sequence. Efficient because we use the Markov property to abstract away previous decisions via this "best score"

This Lecture

- Constituency formalism
- Context-free grammars and the CKY algorithm
- Refining grammars
- Dependency grammar

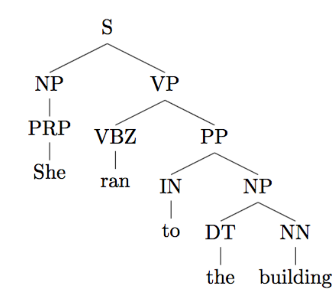
Constituency

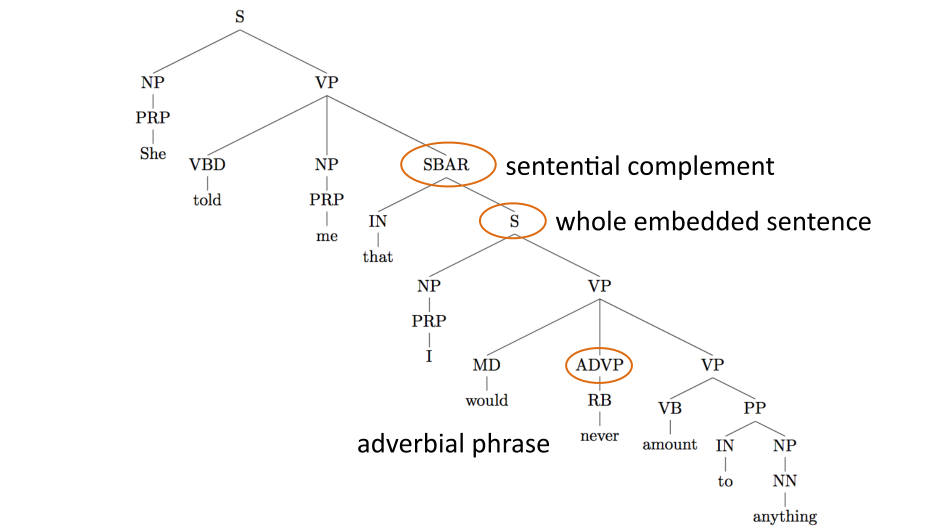
Syntax

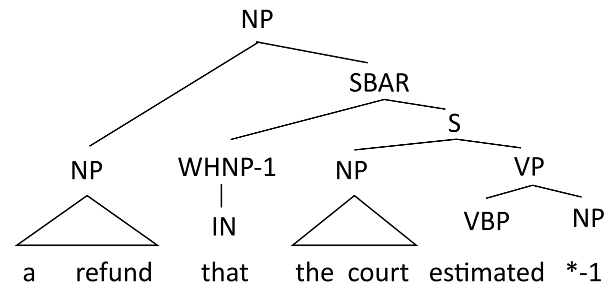
- Study of word order and how words form sentences
- Why do we care about syntax?
  - Multiple interpretations of words (noun or verb?)
  - Recognize verb-argument structures (who is doing what to whom?)
  - Higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV, parsing can canonicalize

Constituency Parsing

- Tree-structured syntactic analyses of sentences
- Common things: noun phrases, verb phrases, prepositional phrases
- Bottom layer is POS tags
- Examples will be in English. Constituency makes sense for a lot of languages but not all

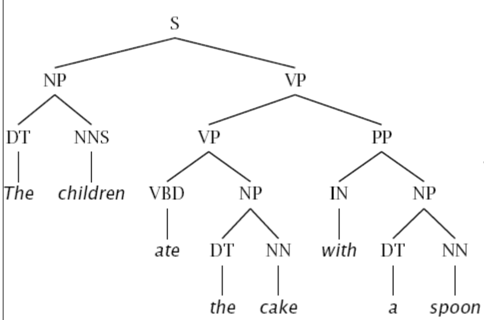


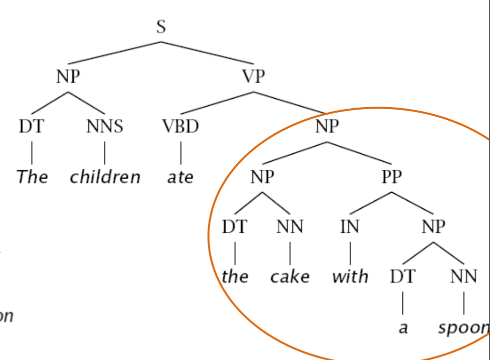




Challenges

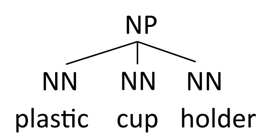
- PP attachment

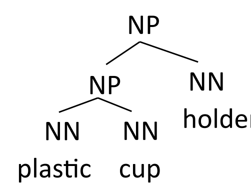




same parse as "the cake with some icing"

Challenges: NP Internal Structure



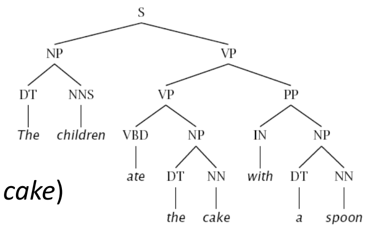


GD What is a plastic cup holder?

A plastic cup holder is a device that is designed to hold cups or other drinking containers, such as cans or bottles. It is typically made of plastic and is often found in vehicles, such as cars or boats, as well as in other settings, such as stadiums or movie theaters.

Constituency

- How do we know what the constituents are?
- Constituency tests:
  - Substitution by *proform* (e.g., pronoun)
  - Clefting (*It was with a spoon that…*)
  - Answer ellipsis (What did they eat? *the cake*)
    (How? *with a spoon*)



- Sometimes constituency is not clear, e.g., coordination: *she went to and bought food at the store*

Context-Free Grammars, CKY

CFGs and PCFGs

Grammar (CFG)

ROOT → S	1.0	NP → NP PP	0.3
S → NP VP	1.0	VP → VBP NP	0.7
NP → DT NN	0.2	VP → VBP NP PP	0.3
NP → NN NNS	0.5	PP → IN NP	1.0

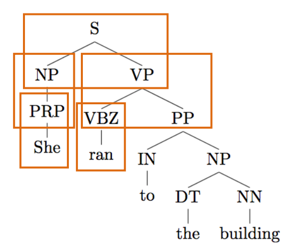
Lexicon

NN → interest	1.0
NNS → raises	1.0
VBP → interest	1.0
VBZ → raises	1.0

- Context-free grammar: symbols which rewrite as one or more symbols
- Lexicon consists of "preterminals" (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
- PCFG: probabilities associated with rewrites, normalize by source symbol

Estimating PCFGs

- Tree *T* is a series of rule applications *r*. $P(T) = \prod_{r \in T} P(r|\text{parent}(r))$

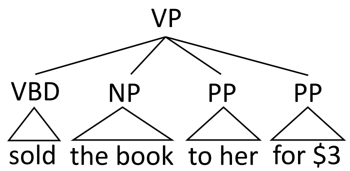


S → NP VP	1.0
NP → PRP	0.5
NP → DT NN	0.5
…	

- Maximum likelihood PCFG for a set of labeled trees: count and normalize! Same as HMMs / Naive Bayes

Binarization

- To parse efficiently, we need our PCFGs to be at most binary (not CNF)

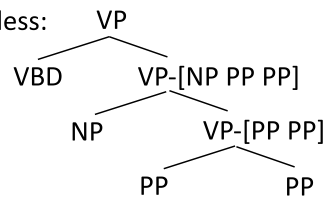


$P(VP \rightarrow VBD\ NP\ PP\ PP) = 0.2$

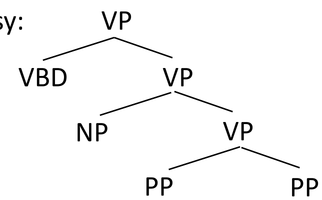
$P(VP \rightarrow VBZ\ PP) = 0.1$

...

- Lossless:

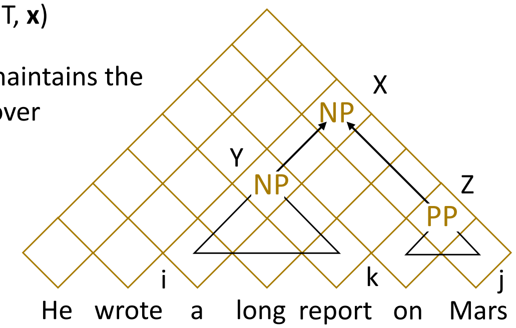


- Lossy:



CKY

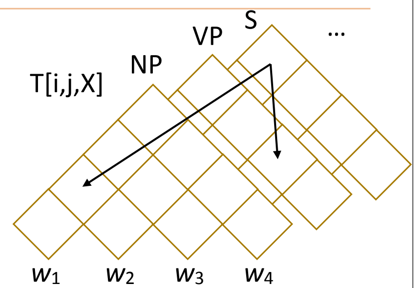
- Find argmax P(T|**x**) = argmax P(T, **x**)
- Dynamic programming: chart maintains the best way of building symbol X over span (i, j)
- CKY = Viterbi, there is also an algorithm called inside-outside = forward-backward



Cocke-Kasami-Younger

CKY

- Chart: T[i,j,X] = best score for X over (i, j)
- Base: $T[i,i+1,X] = \log P(X \rightarrow w_i)$
- Loop over all split points k, apply rules X -> Y Z to build X in every possible way



S[0,4] => NP[0,2] VP[2,4]

- Recurrence:

$$T[i,j,X] = \max_k \max_{r: X \rightarrow X1\ X2} T[i,k,X1] + T[k,j,X2] + \log P(X \rightarrow X1\ X2)$$

- Runtime: $O(n^3 G)$ G = grammar constant

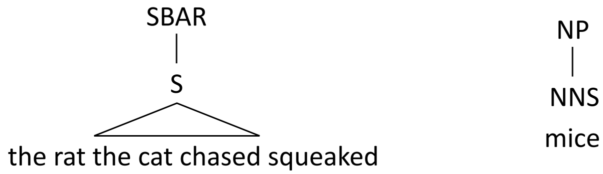
CKY Example

the	child	raises	it
DT -> the 1 NN -> child 1 NNS -> raises 1	VBZ -> raises 1 PRP -> it 1	S -> NP VP 1 NP -> DT NN 1/2 NP -> NN NNS 1/2	VP -> VBZ PRP 1

Recurrence:

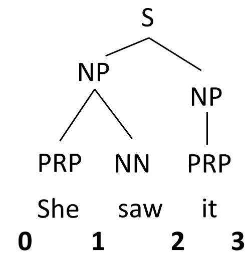
$$T[i,j,X] = \max_k \max_{r: X \rightarrow X1\ X2} T[i,k,X1] + T[k,j,X2] + \log P(X \rightarrow X1\ X2)$$

Unary Rules

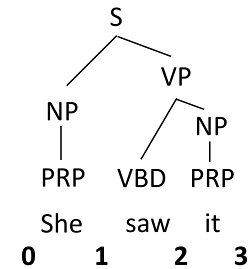


- Unary productions in treebank need to be dealt with by parsers
- Binary trees over n words have at most n-1 nodes, but you can have unlimited numbers of nodes with unaries (S → SBAR → NP → S → ...)
- In practice: enforce at most one unary over each span, modify CKY accordingly

Parser Evaluation



S(0,3),
NP(0,2),
NP(2,3),
~~PRP(0,1),~~
~~NN(1,2),~~
~~PRP(2,3)~~



S(0,3),
NP(0,1),
VP(1,3),
NP(2,3),
~~PRP(0,1),~~
~~VBD(1,2),~~
~~PRP(2,3)~~

- Precision: number of correct brackets / num pred brackets = 2/3
- Recall: number of correct brackets / num of gold brackets = 2/4
- F1: harmonic mean of precision and recall = 0.57

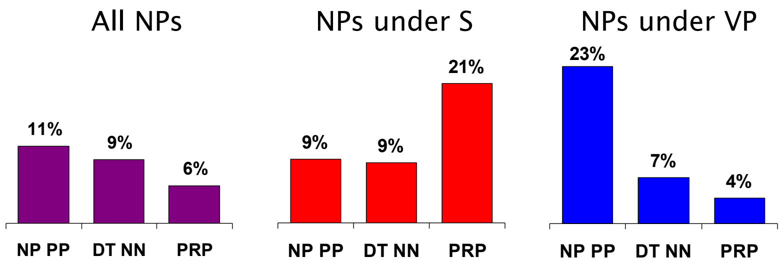
Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
  - Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best PCFGs for English: ~90 F1
- SOTA (discriminative models): 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

Klein and Manning (2003)

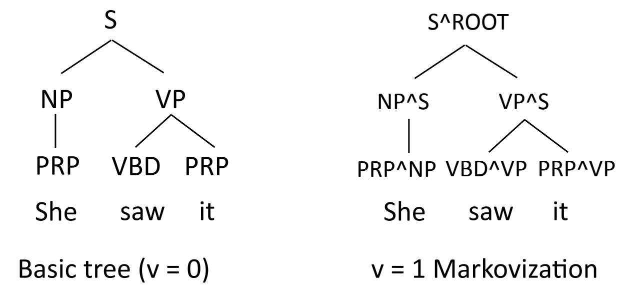
Refining Generative Grammars

PCFG Independence Assumptions



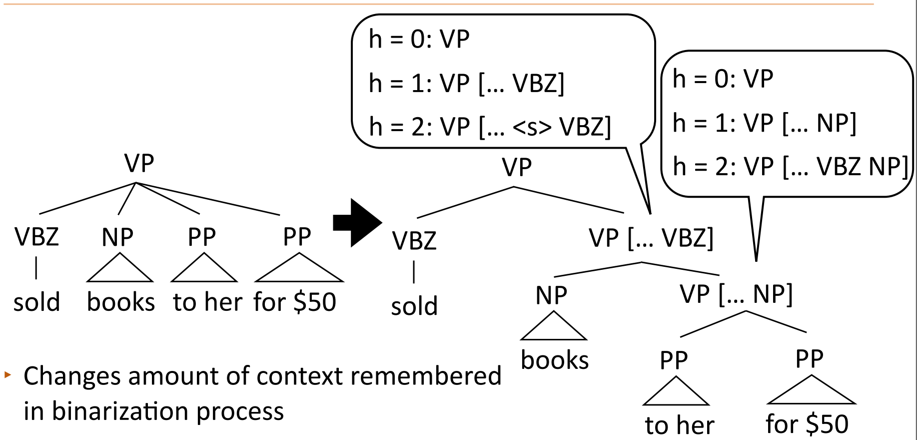
- Language is not context-free: NPs in different contexts rewrite differently
- Can we make the grammar “less context-free”?

Vertical Markovization



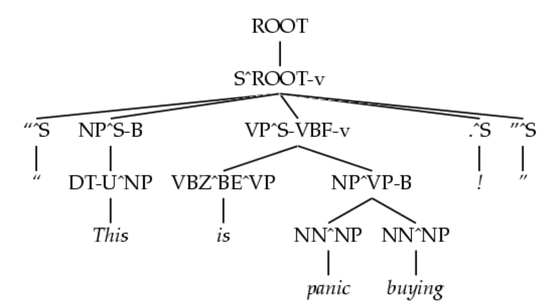
- Why is this a good idea?

Horizontal Markovization



- Changes amount of context remembered in binarization process

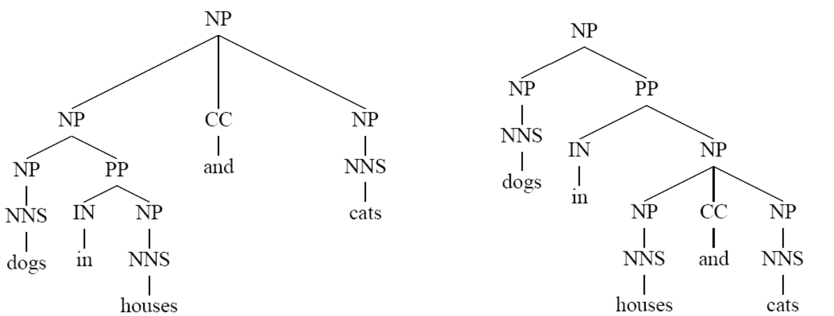
Annotated Tree



- 75 F1 with basic PCFG => 86.3 F1 with this highly customized PCFG, including other tweaks (SOTA was 90 F1 at the time, but with more complex methods)

Klein and Manning (2003)

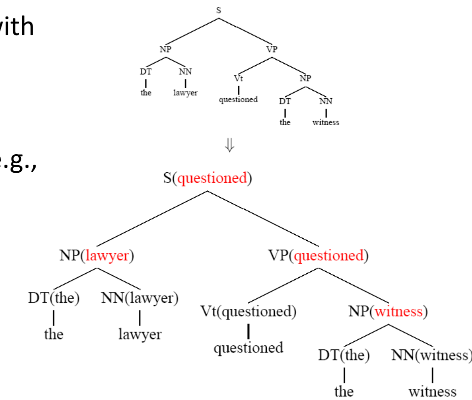
Lexicalized Parsers



- Even with parent annotation, these trees have the same rules. Need to use the words

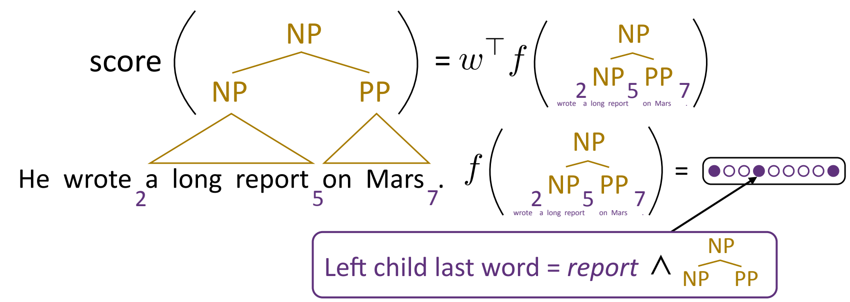
Lexicalized Parsers

- Annotate each grammar symbol with its "head word": most important word of that constituent
- Rules for identifying headwords (e.g., the last word of an NP before a preposition is typically the head)
- Collins and Charniak (late 90s): ~89 F1 with these



State-of-the-art Constituency Parsers

CRF Parsing

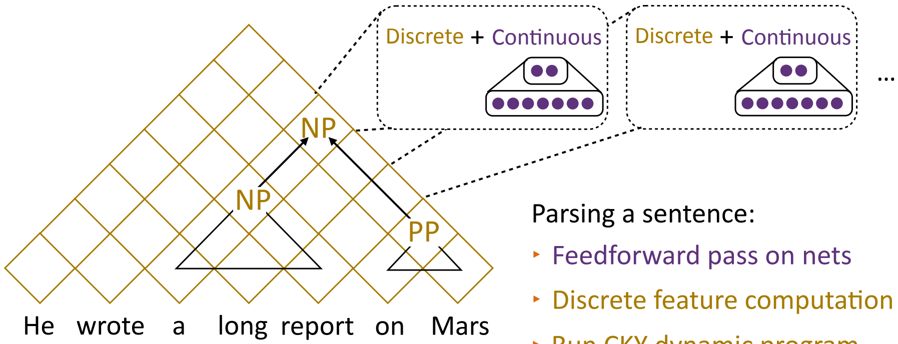


- Can learn that we *report* [PP], which is common due to *reporting on* things
- Can "neuralize" this as well like neural CRFs for NER

Taskar et al. (2004)
Hall, Durrett, and Klein (2014)
Durrett and Klein (2015)

Joint Discrete and Continuous Parsing

- Chart remains discrete!



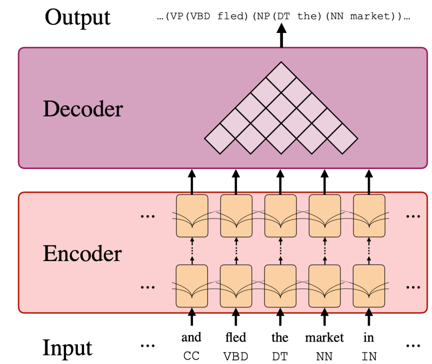
Parsing a sentence:

- Feedforward pass on nets
- Discrete feature computation
- Run CKY dynamic program

Durrett and Klein (ACL 2015)

Pre-trained Models

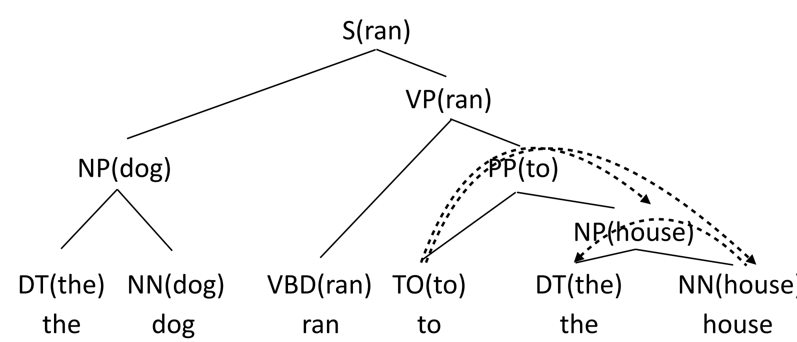
- Improves the neural CRF by using a transformer layer (self-attentive), character-level modeling, and ELMo
- 95.21 on Penn Treebank dev set — much better than past parsers! (~92-93)
- This constituency parser with BERT is one of the strongest today, or use a transition-based version due to Kitaev and Klein (2020)



Kitaev and Klein (2018)

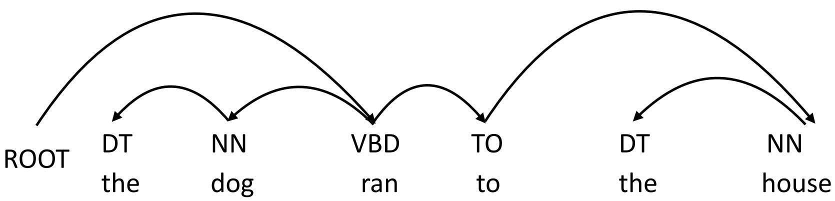
Dependency Syntax

Lexicalized Parsing



Dependency Parsing

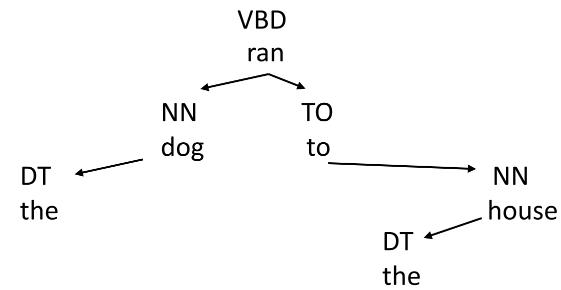
- Dependency syntax: syntactic structure is defined by these arcs
  - Head (parent, governor) connected to dependent (child, modifier)
  - Each word has exactly one parent except for the ROOT symbol, dependencies must form a directed acyclic graph



- POS tags same as before, usually run a tagger first as preprocessing

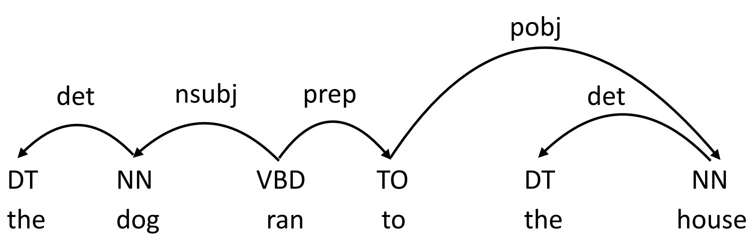
Dependency Parsing

- Still a notion of hierarchy! Subtrees often align with constituents



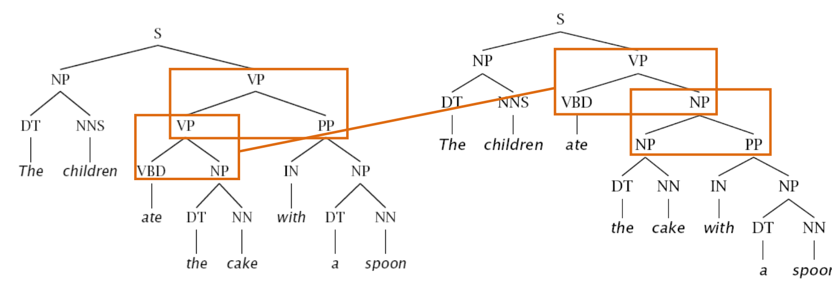
Dependency Parsing

- Can label dependencies according to syntactic function
- Major source of ambiguity is in the structure, so we focus on that more (labeling separately with a classifier works pretty well)



Dependency vs. Constituency: PP Attachment

- Constituency: several rule productions need to change



Dependency vs. Constituency: PP Attachment

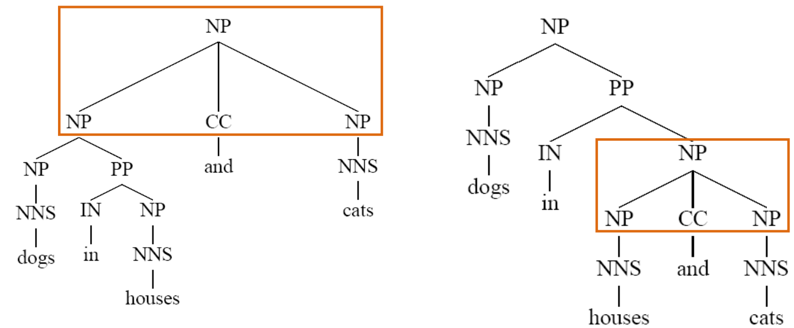
- Dependency: one word (with) assigned a different parent

the children ate the cake with a spoon

- More predicate-argument focused view of syntax
- "What's the main verb of the sentence? What is its subject and object?" — easier to answer under dependency parsing

Dependency vs. Constituency: Coordination

- Constituency: ternary rule NP -> NP CC NP



Dependency vs. Constituency: Coordination

- Dependency: first item is the head

dogs in houses **and cats** dogs in **houses and cats**

[dogs in houses] and cats dogs in [houses and cats]

- Coordination is decomposed across a few arcs as opposed to being a single rule production as in constituency
- Can also choose *and* to be the head
- In both cases, headword doesn't really represent the phrase — constituency representation makes more sense

Takeaways

- PCFGs estimated generatively can perform well if sufficiently engineered
- Neural CRFs work well for constituency parsing
- Next time: revisit lexicalized parsing as *dependency parsing*