# Parser Evaluation

## Parser Evaluation

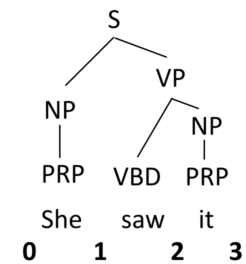
- View a parse as a set of labeled *brackets* / constituents

S(0,3)

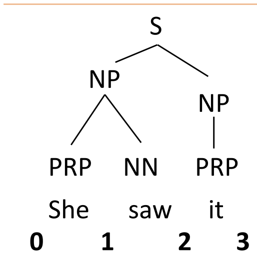
NP(0,1)

PRP(0,1) (but standard evaluation *does not count POS tags*)

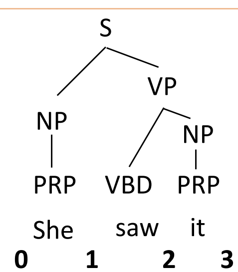
VP(1,3), VBD(1,2), NP(2,3), PRP(2,3)




## 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 predictions / number of predictions = 2/3
- Recall: number of correct predictions / number of golds = 2/4
- F1: harmonic mean of precision and recall = $(1/2 * ((2/4)^{-1} + (2/3)^{-1}))^{-1}$

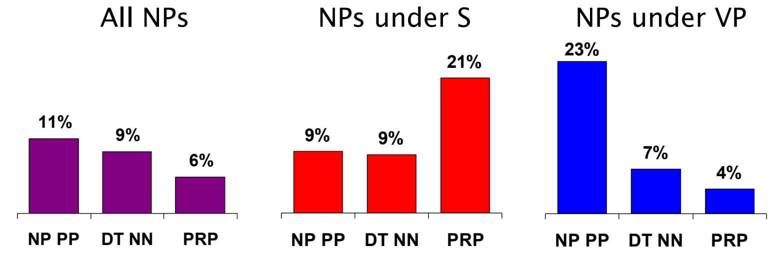
  = 0.57 (closer to min)

## Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
- "Vanilla" PCFG: ~71 F1
- Best PCFGs for English: ~90 F1
- State-of-the-art discriminative models (using unlabeled data): 95 F1
- Other languages: results vary widely depending on annotation + complexity of the grammar

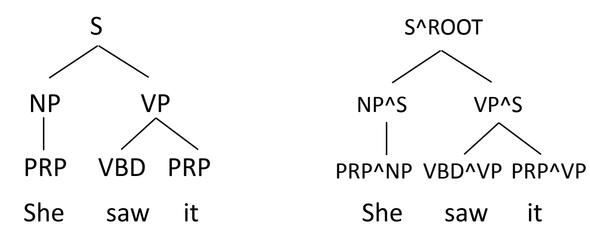
# Refining Generative Grammars

## PCFG Independence Assumptions




- Language is not context-free: NPs in different contexts rewrite differently
- [They]$_{NP}$ received [the package of books]$_{NP}$

## Vertical Markovization



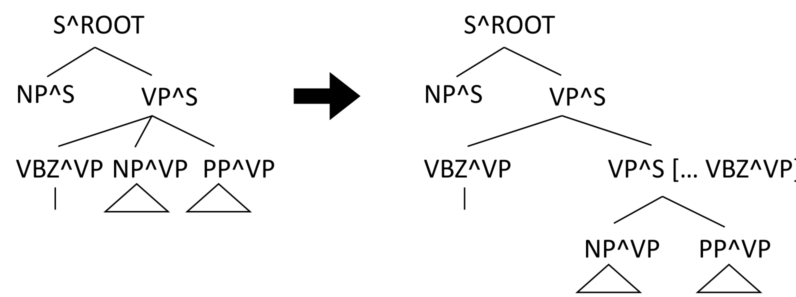

Basic tree (v = 1)

v = 2 Markovization

- Why is this a good idea?

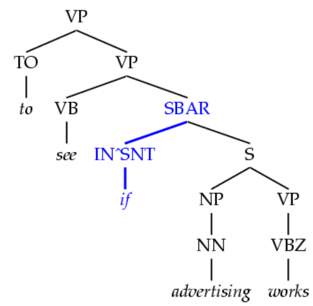
## Annotating Trees

- First apply vertical Markovization, then do another transformation during binarization



## Tag Splits

- Can do some other specialized tag splits: e.g., sentential prepositions behave differently from other prepositions
- ~70 F1 => 86.3 F1 using these tricks

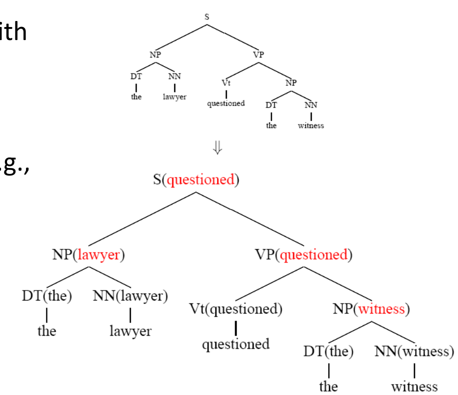



Klein and Manning (2003)

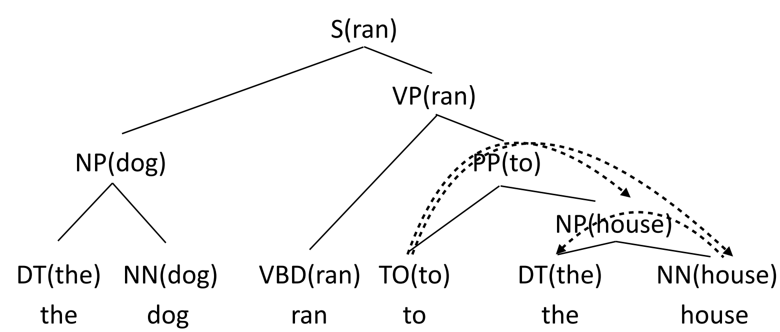
## Lexicalized Parsing, Dependency Parsing

## 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

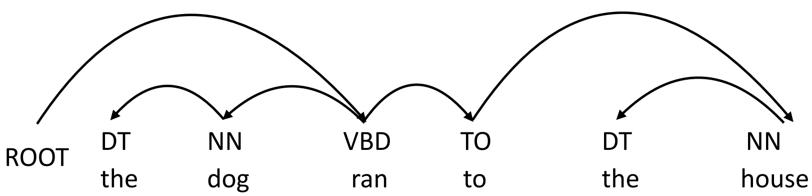



## Lexicalized Parsing



## 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



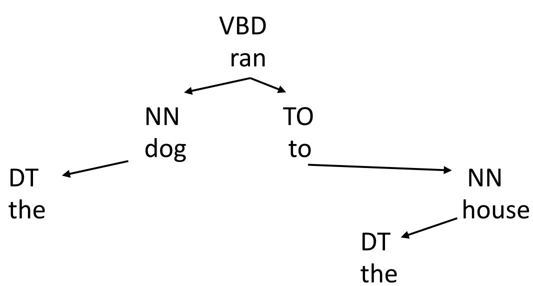

- POS tags same as before, usually run a tagger first as preprocessing

## Why are they defined this way?

- Constituency tests:
  - Substitution by *proform*: the dog *did so [ran to the house]*, *he [the dog]* ran to the house
  - Clefting (*It was [to the house] that the dog ran…*)
- Dependency: verb is the root of the clause, everything else follows from that
  - No notion of a VP!

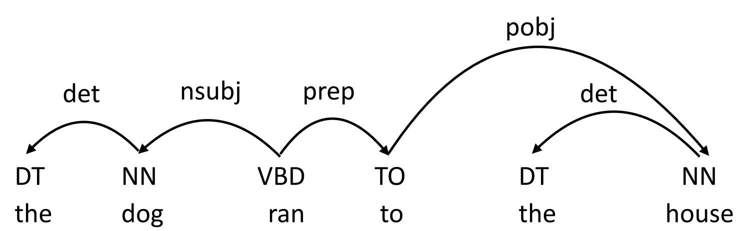
## Dependency Parsing

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



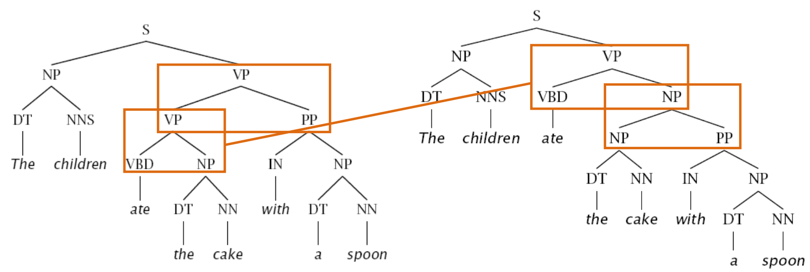

## Dependency Parsing

- 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)



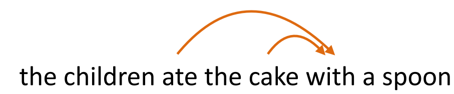
## Dependency vs. Constituency: PP Attachment

- Constituency: several rule productions need to change




## Dependency vs. Constituency: PP Attachment

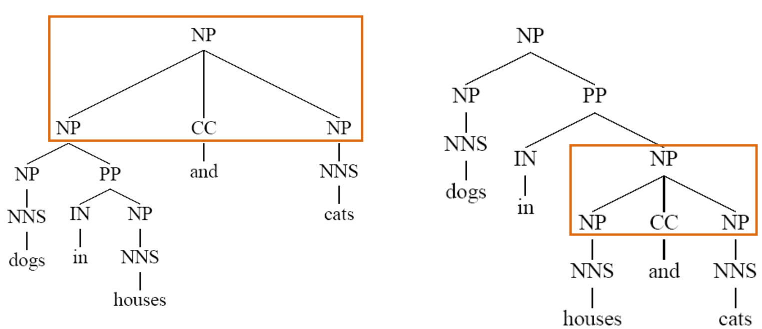
- Dependency: one word (with) assigned a different parent




- 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

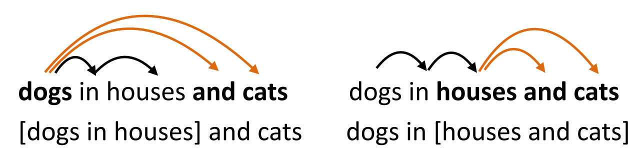
## Dependency vs. Constituency: Coordination

- Constituency: ternary rule NP -> NP CC NP




## Dependency vs. Constituency: Coordination

- Dependency: first item is the head




- 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

Shift-Reduce Parsing
(see notes)