Lecture 16:
Probabilistic CFG Parsing

Intro to NLP, CS585, Fall 2014

http://people.cs.umass.edu/~brenocon/inlp2014/

Brendan O'Connor

CMPSCI 585 In-class Exercise 10/28 (actually 10/30) Name: ____________________

Fill in the CYK dynamic programming table to parse the sentence below. In the bottom right corner, draw the two parse trees.

	she	eats	fish	with	chop-sticks
0	1	2	3	4	5

	1	2	3	4	5
0	NP				
1					
2					
3					
4					

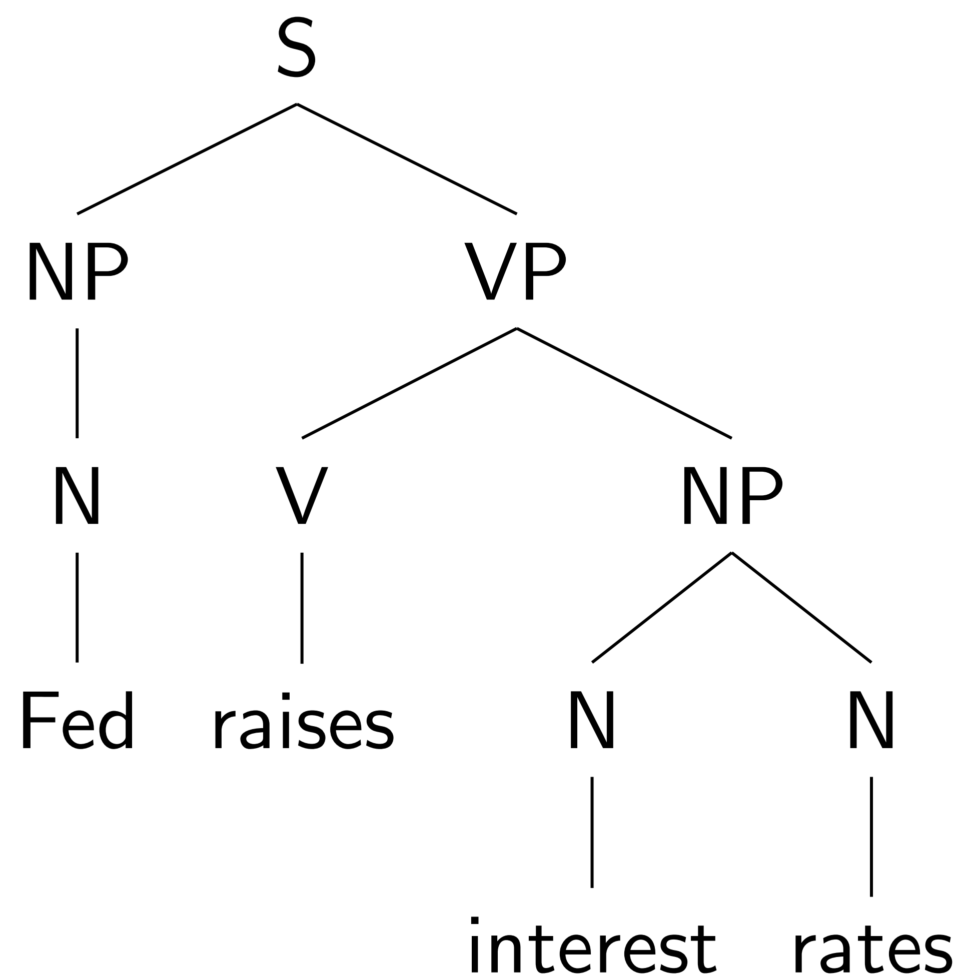
S → NP VP
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP

NP → she
NP → fish
NP → fork
NP → chopsticks
V → eats
V → fish
P → with

- (Solution slide removed for web; see the piazza resources page)

- OK, we can track ambiguities. But how to resolve them?
- Need to *prefer* certain trees/derivations to others.

Another example



- A minimal grammar permits 36 parses!
- Broad-coverage grammars permit millions of parses of moderate-size sentences.

[Slide: Jacob Eisenstein]

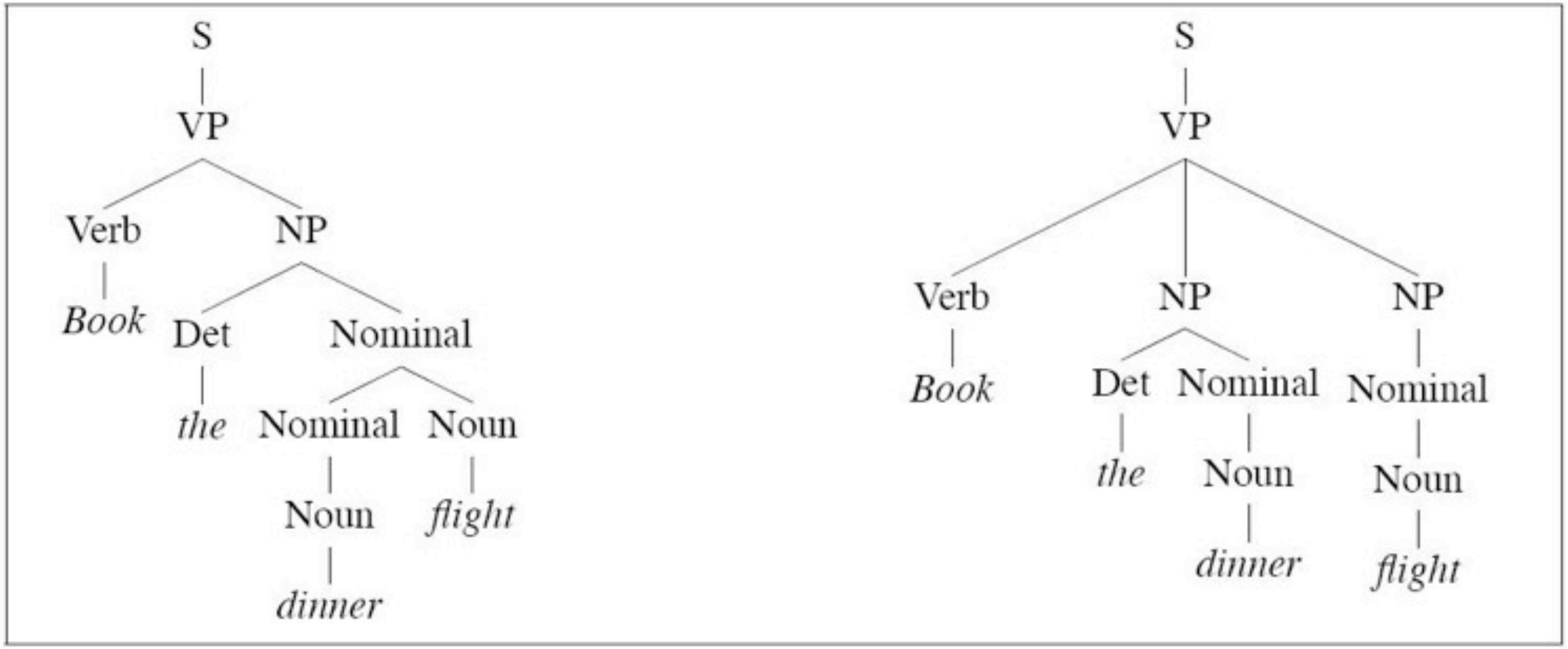
PCFGs

S	$\rightarrow$ NP VP	0.9
S	$\rightarrow$ S CC S	0.1
NP	$\rightarrow$ N	0.2
NP	$\rightarrow$ DT N	0.3
NP	$\rightarrow$ N NP	0.2
NP	$\rightarrow$ JJ NP	0.2
NP	$\rightarrow$ NP PP	0.1
VP	$\rightarrow$ V	0.4
VP	$\rightarrow$ V NP	0.3
VP	$\rightarrow$ V NP NP	0.1
VP	$\rightarrow$ VP PP	0.2
PP	$\rightarrow$ P NP	1.0

- P(words, tree) = product of all expansion probs
- For each nonterminal, possible expansions sum to 1

$$P(\text{tree} \mid \text{words}) = \frac{1}{Z} P(\text{tree}, \text{words})$$

$$P(\text{tree}, \text{words}) = \text{product of all expansion probs}$$



Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flights	.40	Noun	→	flights	.40

Major Research Questions

✓ What's the right **representation**?
✓ What's the right **model**?

(We've talked about one representation and one model.)

- How to learn to parse **empirically**?
- How to make parsers **fast**?
- How to incorporate structure **downstream**?

[Slides: Noah Smith]

Decoding Algorithms

- Suppose I have a PCFG and a sentence.
- What might I want to do?
  - Find the most likely tree (if it exists).
  - Find the *k* most likely trees.
  - Gather statistics on the **distribution** over trees.

- Should remind you of FS models!

[Slides: Noah Smith]

Probabilistic CKY

Input: PCFG G = (Σ, **N**, S, **R**) in CNF and sequence **w** ∈ **Σ***

Output: most likely tree for **w**, if it exists, and its probability.

$$C(X,i,i) = \langle p(X \to w_i), \text{null} \rangle$$

$$C(X,i,j) = \left\langle \begin{array}{l} \max\limits_{Y,Z \in \mathbf{N}, k \in [i+1, j-2]} C(Y,i,k) \cdot C(Z,k+1,j) \cdot p(X \to Y,Z), \\ \& \operatorname*{arg\,max}\limits_{Y,Z \in \mathbf{N}, k \in [i+1, j-2]} C(Y,i,k) \cdot C(Z,k+1,j) \cdot p(X \to Y,Z) \end{array} \right\rangle$$

$$\text{goal} = C(S,1,|\mathbf{w}|)$$

[Slides: Noah Smith]

Resist This Temptation!

- CKY is not “building a tree” bottom-up.
- It is scoring partial hypotheses bottom-up.
- You can assume nothing about the tree until you get to the end!

[Slides: Noah Smith]

HMM and PCFGs

- PCFGs are a generalization of HMMs

	Sequence	Tree
Decoding	Viterbi	CKY
Decoding Complexity	linear in sent. length	cubic in sent. length

Learning from Data

1. Where do the **rules** come from?
2. Where do the rule **probabilities** come from?

First answer: Look at a huge collection of trees (a treebank).

$X \rightarrow \alpha$ is in the grammar iff it's in the treebank.

$p(\alpha \mid X)$ is proportional to the count of $X \rightarrow \alpha$.

[Slides: Noah Smith]

Penn Treebank (Marcus et al. 1993)

- A million tokens of parsed sentences from the Wall Street Journal
  - There's also parses of the Brown corpus -- fiction, essays, etc. -- but researchers usually ignore it
- Parsed by experts (trained annotators), with consensus process for disagreement
- The structure looks like what you'd expect from a PCFG.
  - Traces ... usually ignored by most parsers
  - Tends to be "flat" where there's controversy

Example Tree

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

[Slides: Noah Smith]

```
( (S
    (NP-SBJ-1
      (NP (NNP Rudolph) (NNP Agnew) )
      (, ,)
      (UCP
        (ADJP
          (NP (CD 55) (NNS years) )
          (JJ old) )
        (CC and)
        (NP
          (NP (JJ former) (NN chairman) )
          (PP (IN of)
            (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))
      (, ,) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate)
   ))
))))
    (. .) ))
```

[Slides: Noah Smith]

Evaluating Parsers

- Take a sentence from the test set.
- Use your parser to propose a **hypothesis** parse.
- Treebank gives you the **correct** parse.
- How to compare?
  - {unlabeled, labeled} × {precision, recall}
  - crossing brackets statistics
  - `evalb` (http://nlp.cs.nyu.edu/evalb)
- Significance testing …

[Slides: Noah Smith]

Issues

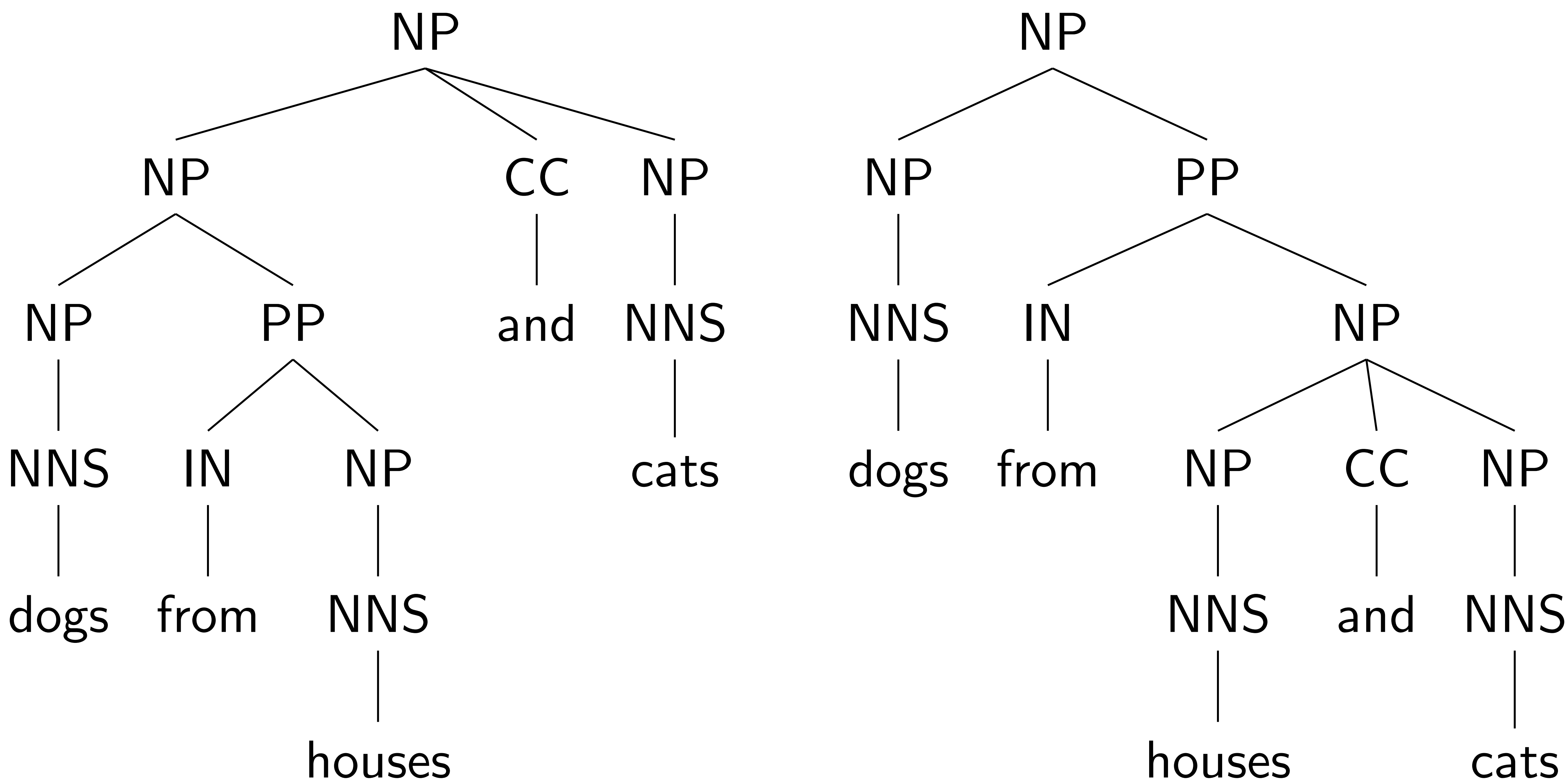
- This same dataset has been intensively used since 1993 for English parsing research
  - Why might this be an issue?
- Treebanks for other languages may require different grammatical conventions; quality varies
- It's pretty easy to find issues in English PTB, though quality seems reasonably high
- Issue: domain transfer

Training Parsers In Practice

- Transformations on trees
  - Some of these are generally taken to be crucial
  - Some are widely debated
  - Lately, people have started **learning** these transformations
- Smoothing (crucial)
- We will come back to this as we explore some current state-of-the art parsers.

[Slides: Noah Smith]

Problems with PCFGs



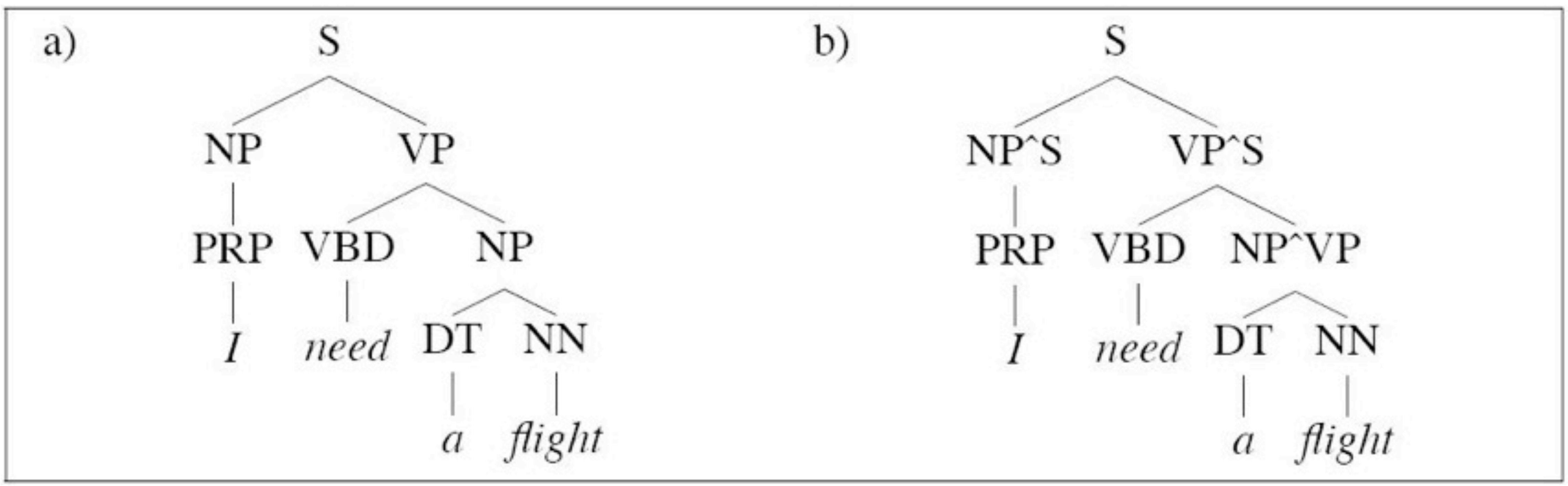
Modern statistical parsers

- PCFG assumptions are too strong.
  How to improve?
  - Transform the training data
    - splitting/“annotating” non-terminals
  - Automatically learn better splits with EM
    (“Berkeley parser”)
  - Discriminative whole-tree features -- typically have to use re-ranking
- Or, shift-reduce parsing: completely alternative approach to constituency parsing
  - Seems to be fastest with best accuracy, right now at least??
  - Zhang’s *zpar,* or a similar one within the Stanford parser software

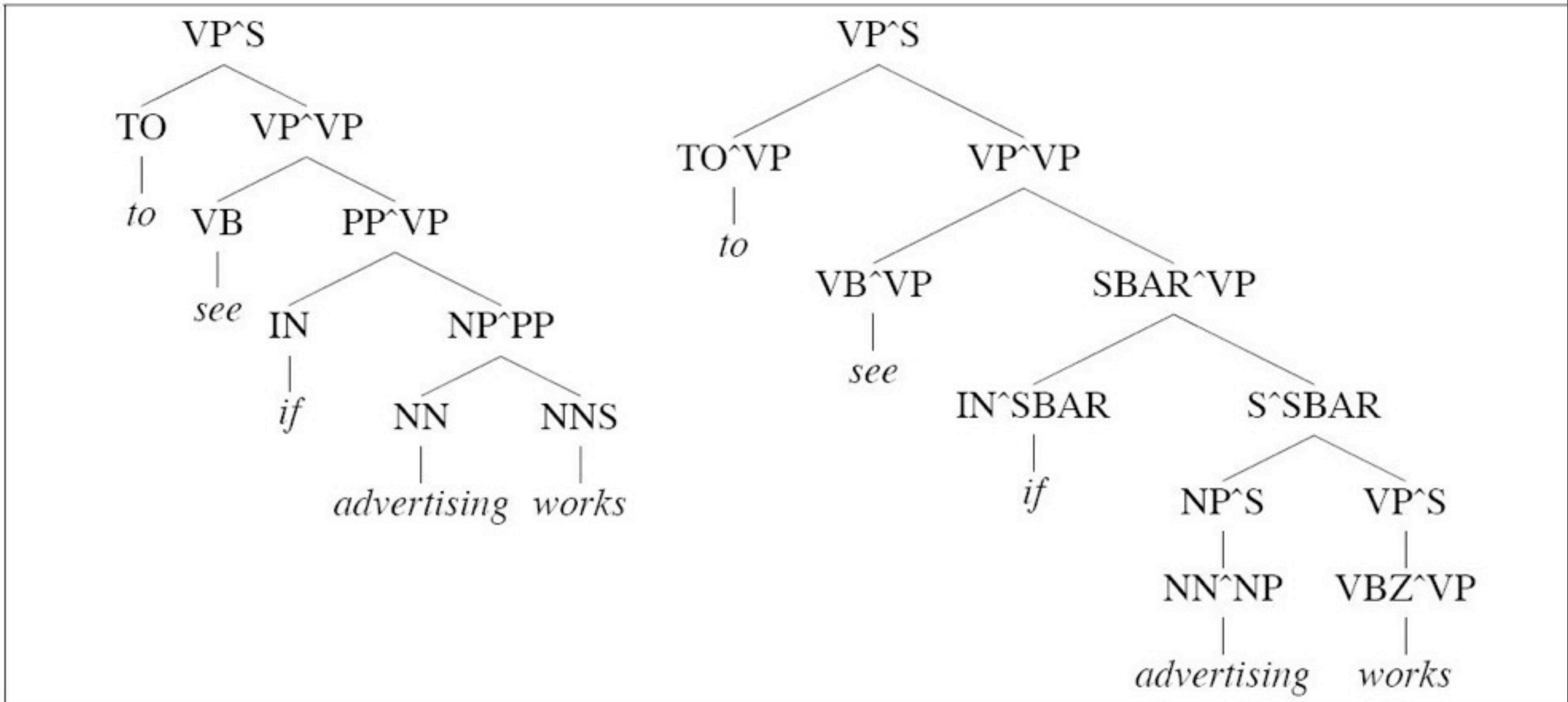
- Next week: direct dependency parsing

Non-terminal splits

- Annotate a nontemrminal symbol its parent/ grandparent/sibling
  - Relaxes PCFG independence assumptions



Non-terminal splits

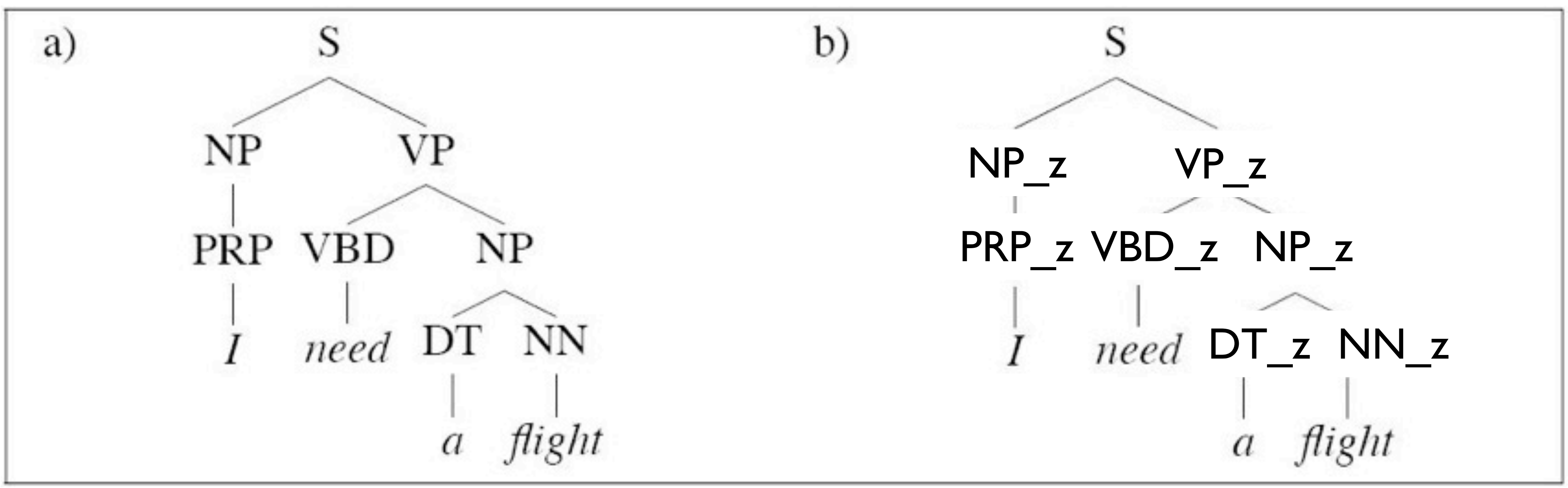


- Left: still incorrect
  Right: split preterminals. "if" prefers to be sentential complement.

- stopped here

Latent-variable PCFG

- Want to automatically learn the splits!
- Latent-variable PCFG: augment training data with latent states. Learn with EM. Use "split-merge" training to vary number of latent states.
    - NP_1, NP_2, NP_3....
- [Petrov (2009), still used today in open-source Berkeley parser]



Discriminative re-ranking

- Take top-K trees from a PCFG.
- Re-rank them with log-linear model that can use *whole-tree* features: e.g. "does this NP contain 15-20 words"?
  - This model is more powerful than a PCFG.
  - But by itself, inference is intractable.
- BLIPP parser [Charniak and Johnson 2005]: might still be the most accurate English parser
- Re-ranking is a very powerful general technique in NLP

How good are parsers now?

- Labeled precision/recall: 90-93% F1 score
- Whole tree accuracy: much less!
- Which ambiguities or errors matter for what types of tasks?