CS447: Natural Language Processing

http://courses.engr.illinois.edu/cs447

Lecture 17: More on PCFG parsing

Julia Hockenmaier

juliahmr@illinois.edu

3324 Siebel Center

Probabilistic Context-Free Grammars

For every nonterminal X, define a probability distribution $P(X \rightarrow \alpha \mid X)$ over all rules with the same LHS symbol X:

S	$\longrightarrow$	NP VP	0.8
S	$\longrightarrow$	S conj S	0.2
NP	$\longrightarrow$	Noun	0.2
NP	$\longrightarrow$	Det Noun	0.4
NP	$\longrightarrow$	NP PP	0.2
NP	$\longrightarrow$	NP conj NP	0.2
VP	$\longrightarrow$	Verb	0.4
VP	$\longrightarrow$	Verb NP	0.3
VP	$\longrightarrow$	Verb NP NP	0.1
VP	$\longrightarrow$	VP PP	0.2
PP	$\longrightarrow$	P NP	1.0

Transforming a PCFG to Chomsky Normal Form

S	$\longrightarrow$	NP VP	0.8
S	$\longrightarrow$	S conj S	0.2
NP	$\longrightarrow$	Noun	0.2
NP	$\longrightarrow$	Det Noun	0.4
NP	$\longrightarrow$	NP PP	0.2
NP	$\longrightarrow$	NP conj NP	0.2
VP	$\longrightarrow$	Verb	0.3
VP	$\longrightarrow$	Verb NP	0.3
VP	$\longrightarrow$	Verb NP NP	0.1
VP	$\longrightarrow$	VP PP	0.3
PP	$\longrightarrow$	Prep NP	1.0
Prep	$\longrightarrow$	P	1.0
Noun	$\longrightarrow$	N	1.0
Verb	$\longrightarrow$	V	1.0

This grammar is not in Chomsky Normal Form:

- Ternary rules, e.g:
  S $\longrightarrow$ S conj S
- RHS with nonterminals and terminals
  S $\longrightarrow$ S conj S
- Unary rules from one nonterminal to another, e.g.:
  VP $\longrightarrow$ Verb

Transforming a PCFG to Chomsky Normal Form

S	⟶	NP VP	0.8
S	⟶	S conj S	0.2
NP	⟶	Noun	0.2
NP	⟶	Det Noun	0.4
NP	⟶	NP PP	0.2
NP	⟶	NP conj NP	0.2
VP	⟶	Verb	0.3
VP	⟶	Verb NP	0.3
VP	⟶	Verb NP NP	0.1
VP	⟶	VP PP	0.3
PP	⟶	Prep NP	1.0
Prep	⟶	P	1.0
Noun	⟶	N	1.0
Verb	⟶	V	1.0

S	⟶	**NP VP**	**0.8**
S	⟶	**S conjS**	**0.2**
conjS	⟶	**Conj S**	**1.0**
NP	⟶	**N**	**0.2**
NP	⟶	**Det Noun**	**0.4**
NP	⟶	**NP PP**	**0.2**
NP	⟶	**NP conjNP**	**0.2**
conjNP	⟶	**Conj NP**	**1.0**
VP	⟶	**V**	**0.3**
VP	⟶	**Verb NP**	**0.3**
VP	⟶	**Verb NPNP**	**0.1**
NPNP	⟶	**NP NP**	**1.0**
VP	⟶	**Verb PP**	**0.3**
PP	⟶	**Prep NP**	**1.0**
Prep	⟶	**P**	**1.0**
Noun	⟶	**N**	**1.0**
Verb	⟶	**V**	**1.0**
Conj	⟶	**C**	**1.0**
Det	⟶	**D**	**1.0**

Transforming a PCFG to Chomsky Normal Form

What we did here:

- New NTs for ternary rules

  S ⟶ S conj S 0.2

  is now

  S ⟶ S conjS 0.2

  conjS ⟶ Conj S 1.0

- Introduced new terminals

  Conj ⟶ C

  Det ⟶ D

- Removed unary rules from one NT to another, e.g.:

  VP ⟶ Verb 0.3

  is now

  VP ⟶ V 0.3

S	⟶	NP VP	0.8
S	⟶	S conjS	0.2
conjS	⟶	Conj S	1.0
NP	⟶	N	0.2
NP	⟶	Det Noun	0.4
NP	⟶	NP PP	0.2
NP	⟶	NP conjNP	0.2
conjNP	⟶	Conj NP	1.0
VP	⟶	V	0.3
VP	⟶	Verb NP	0.3
VP	⟶	Verb NPNP	0.1
NPNP	⟶	NP NP	1.0
VP	⟶	Verb PP	0.3
PP	⟶	Prep NP	1.0
Prep	⟶	P	1.0
Noun	⟶	N	1.0
Verb	⟶	V	1.0
Conj	⟶	C	1.0
Det	⟶	D	1.0

PCFG parsing (decoding): Probabilistic CKY

Probabilistic CKY: Viterbi

Like standard CKY, but with probabilities.
Finding the most likely tree is similar to Viterbi for HMMs:

Initialization:

- [*optional*] Every chart entry that corresponds to a **terminal** (entries w in `cell[i][i]`) has a Viterbi probability $P_{\mathrm{VIT}}(\mathrm{w}_{[i][i]}) = 1$ (*)
- Every entry for a **non-terminal** `X` in `cell[i][i]` has Viterbi probability $P_{\mathrm{VIT}}(\mathrm{X}_{[i][i]}) = \mathrm{P}(\mathrm{X} \rightarrow \mathrm{w} \mid \mathrm{X})$ [and a single backpointer to $\mathrm{w}_{[i][i]}$ (*)]

Recurrence: For every entry that corresponds to a **non-terminal** `X` in `cell[i][j]`, keep only the highest-scoring pair of backpointers to any pair of children (`Y` in `cell[i][k]` and `Z` in `cell[k+1][j]`):

$$P_{\mathrm{VIT}}(\mathrm{X}_{[i][j]}) = \mathrm{argmax}_{\mathrm{Y,Z},k}\ P_{\mathrm{VIT}}(\mathrm{Y}_{[i][k]}) \times P_{\mathrm{VIT}}(\mathrm{Z}_{[k+1][j]}) \times P(\mathrm{X} \rightarrow \mathrm{Y\ Z} \mid \mathrm{X})$$

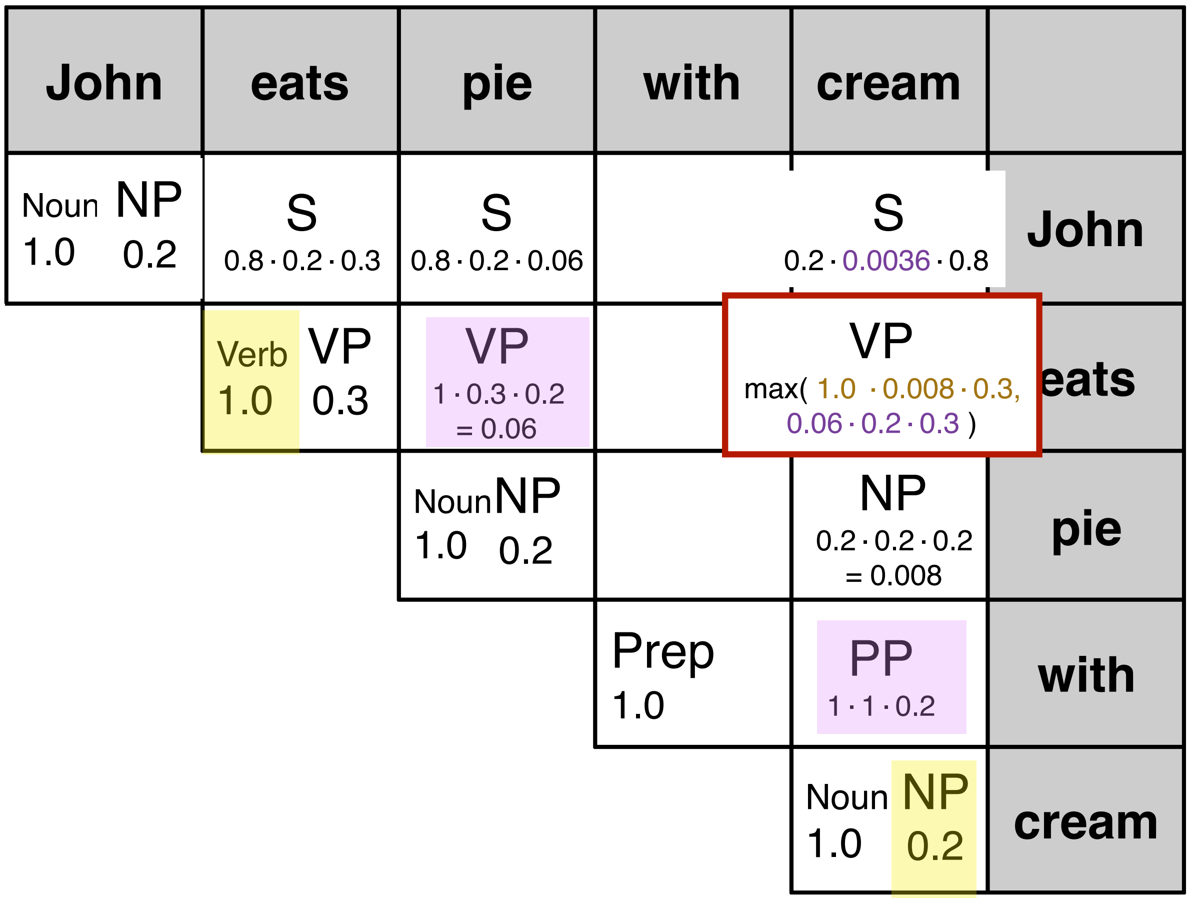
Final step: Return the Viterbi parse for the start symbol S in the top `cell[1][n]`.

*this is unnecessary for simple PCFGs, but can be helpful for more complex probability models

Probabilistic CKY

Input: POS-tagged sentence

`John_N eats_V pie_N with_P cream_N`



S	⟶	NP VP	0.8
S	⟶	S conjS	0.2
conjS	⟶	Conj S	1.0
NP	⟶	N	0.2
NP	⟶	Det Noun	0.4
NP	⟶	NP PP	0.2
NP	⟶	NP conjNP	0.2
conjNP	⟶	Conj NP	1.0
VP	⟶	V	0.3
VP	⟶	Verb NP	0.3
VP	⟶	Verb NPNP	0.1
NPNP	⟶	NP NP	1.0
VP	⟶	Verb PP	0.3
PP	⟶	Prep NP	1.0
Prep	⟶	P	1.0
Noun	⟶	N	1.0
Verb	⟶	V	1.0
Conj	⟶	C	1.0
Det	⟶	D	1.0

How well can a PCFG model the distribution of trees?

PCFGs make **independence assumptions**:

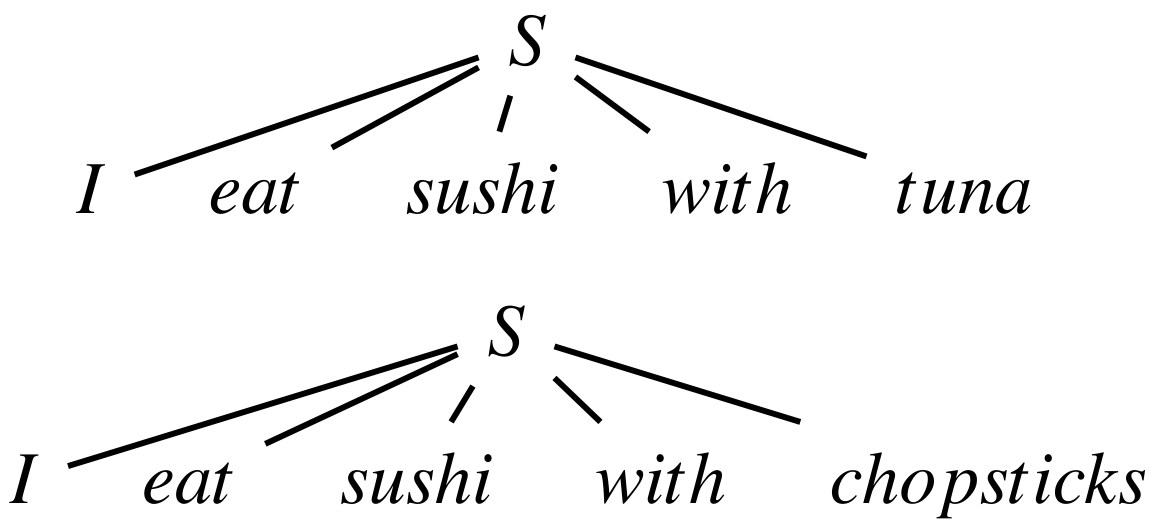
Only the label of a node determines what children it has.

Factors that influence these assumptions:

Shape of the trees:
A corpus with **flat trees** (i.e. few nodes/sentence)
results in a model with few independence assumptions.

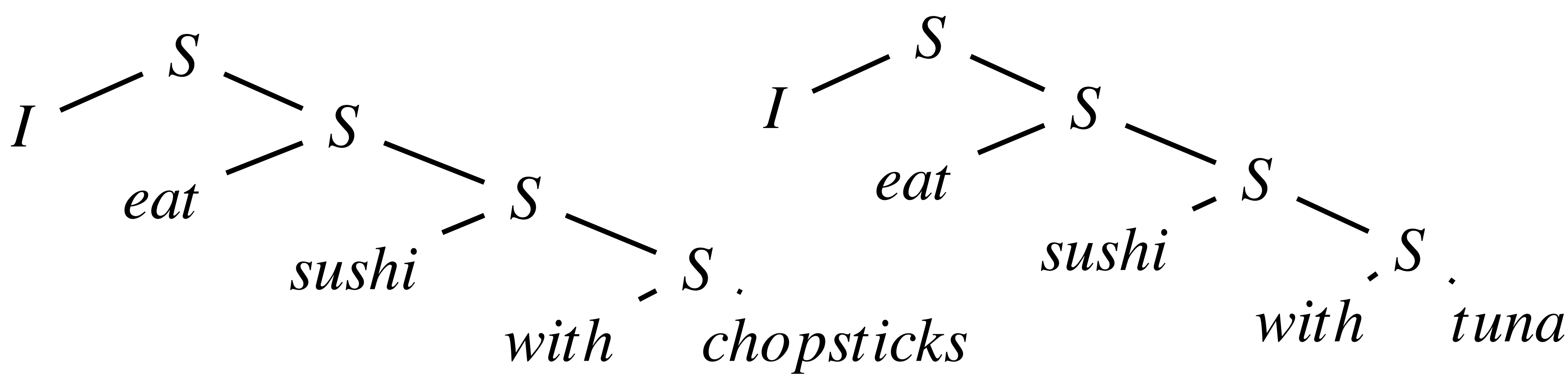
Labeling of the trees:
A corpus with **many node labels** (nonterminals)
results in a model with few independence assumptions.

Example 1: flat trees



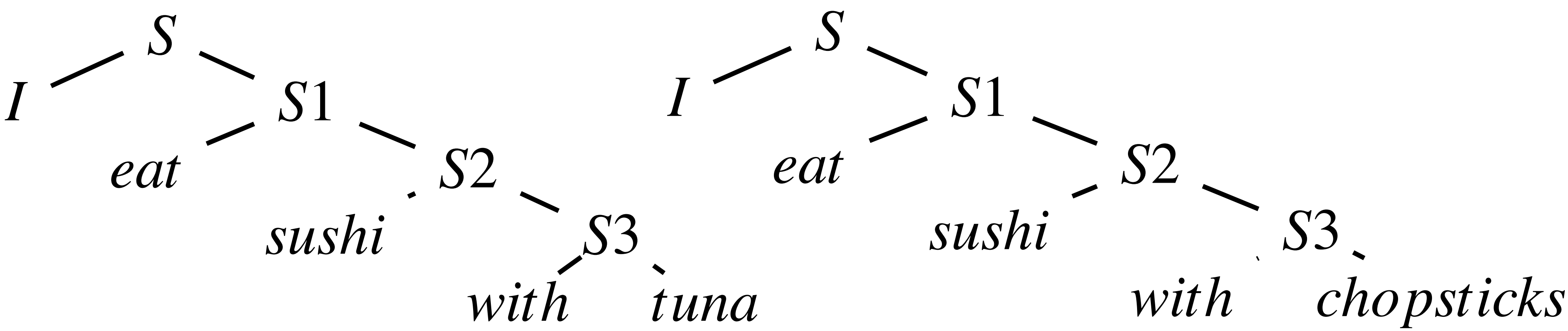
What sentences would a PCFG estimated from this corpus generate?

Example 2: deep trees, few labels



What sentences would a PCFG estimated from this corpus generate?

Example 3: deep trees, many labels



What sentences would a PCFG estimated from this corpus generate?

Aside: Bias/Variance tradeoff

A probability model has low **bias** if it makes
few independence assumptions.
⇒ It can capture the structures in the training data.

This typically leads to a more fine-grained partitioning of the training data.

Hence, fewer data points are available to estimate the model parameters.

This increases the **variance** of the model.
⇒ This yields a poor estimate of the distribution.

Parser evaluation

Precision and recall

Precision and recall were originally developed as evaluation metrics for information retrieval:

- **Precision:** What percentage of retrieved documents are relevant to the query?
- **Recall**: What percentage of relevant documents were retrieved?

In NLP, they are often used in addition to accuracy:

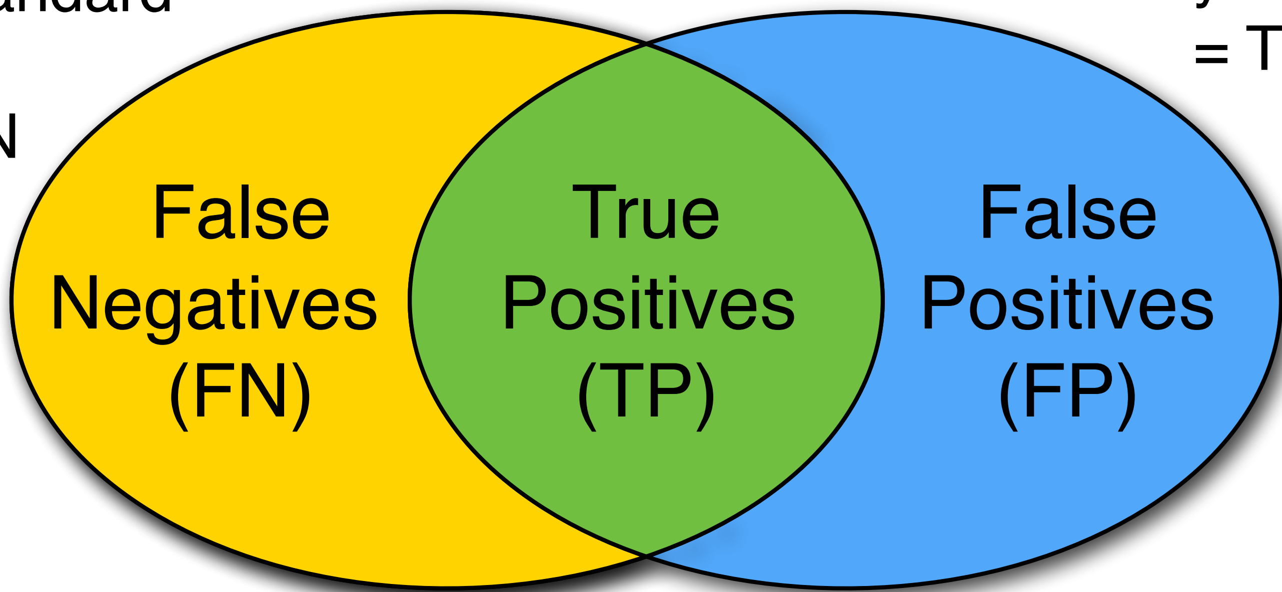
- **Precision:** What percentage of items that were assigned label X do actually have label X in the test data?
- **Recall:** What percentage of items that have label X in the test data were assigned label X by the system?

Particularly useful when there are more than two labels.

True vs. false positives, false negatives

Items labeled X
in the gold standard
(‘truth’)
= TP + FN

Items labeled X
by the system
= TP + FP

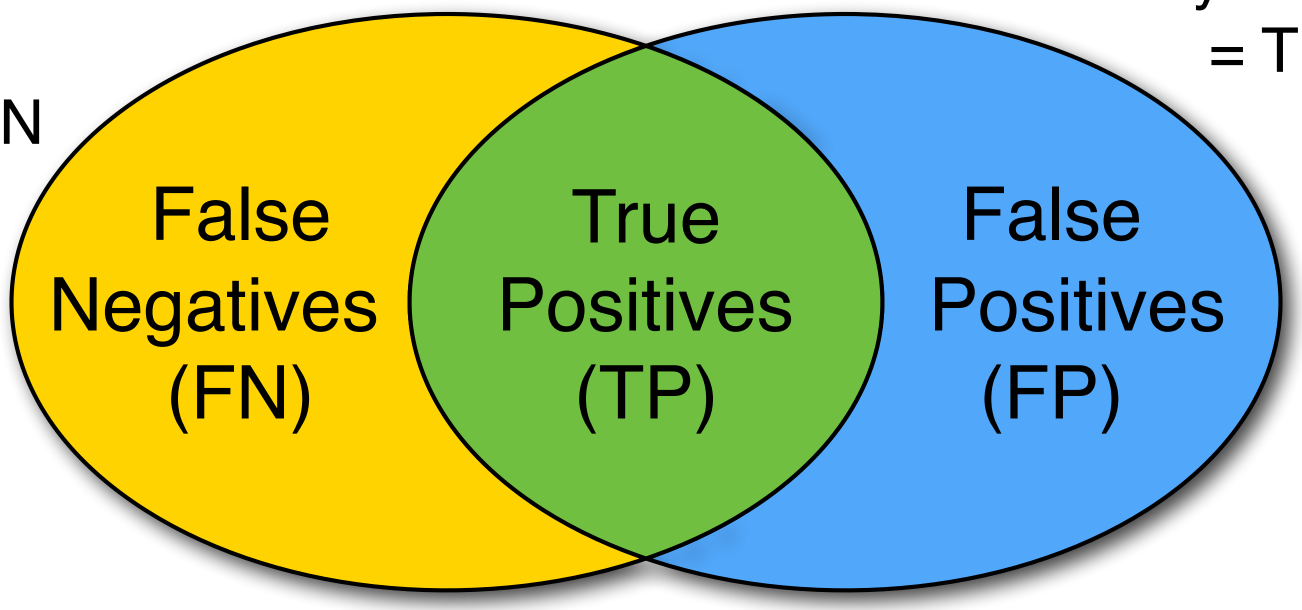


- True positives: Items that were labeled X by the system, and should be labeled X.
- False positives: Items that were labeled X by the system, but should not be labeled X.
- False negatives: Items that were not labeled X by the system, but should be labeled X

Precision, recall, f-measure

Items labeled X
in the gold standard
(‘truth’)
= TP + FN

Items labeled X
by the system
= TP + FP



Precision: P = TP / (TP + FP)

Recall: R = TP / (TP + FN)

F-measure: harmonic mean of precision and recall

$$F = (2 \cdot P \cdot R) / (P + R)$$

Evalb (“Parseval”)

Measures recovery of phrase-structure trees.

Labeled: span and label (NP, PP,...) has to be right.

[Earlier variant— unlabeled**:** span of nodes has to be right]

Two aspects of evaluation

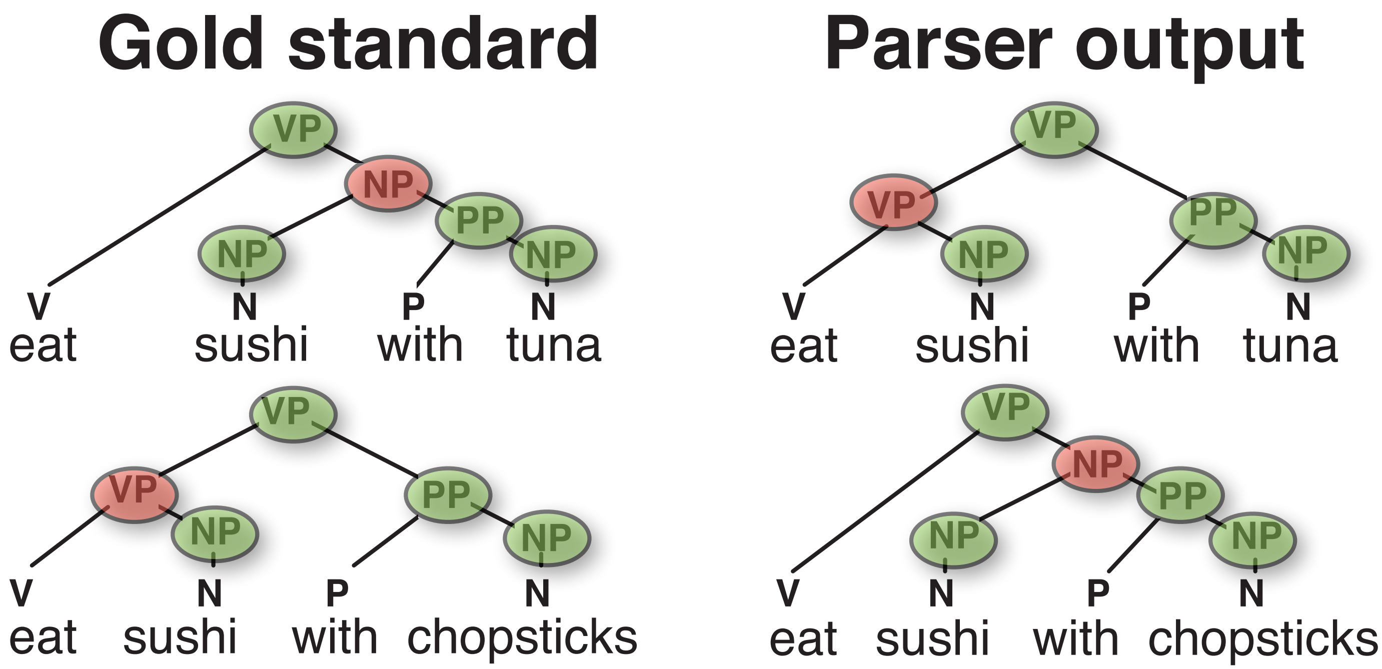
Precision: *How many of the predicted nodes are correct?*

Recall: *How many of the correct nodes were predicted?*

Usually combined into one metric ***(F-measure)****:*

$$
\begin{aligned}
P &= \frac{\#\text{correctly predicted nodes}}{\#\text{predicted nodes}} \\
R &= \frac{\#\text{correctly predicted nodes}}{\#\text{correct nodes}} \\
F &= \frac{2PR}{P+R}
\end{aligned}
$$

Parseval in practice



eat sushi with tuna: Precision: 4/5 Recall: 4/5

eat sushi with chopsticks: Precision: 4/5 Recall: 4/5

Penn Treebank parsing

The Penn Treebank

The first publicly available syntactically annotated corpus

Wall Street Journal (50,000 sentences, 1 million words)

also Switchboard, Brown corpus, ATIS

The annotation:

- POS-tagged (Ratnaparkhi's MXPOST)
- Manually annotated with phrase-structure trees
- Richer than standard CFG: *Traces* and other *null elements* used to represent non-local dependencies (designed to allow extraction of predicate-argument structure) [more on this later in the semester]

Standard data set for English parsers

The Treebank label set

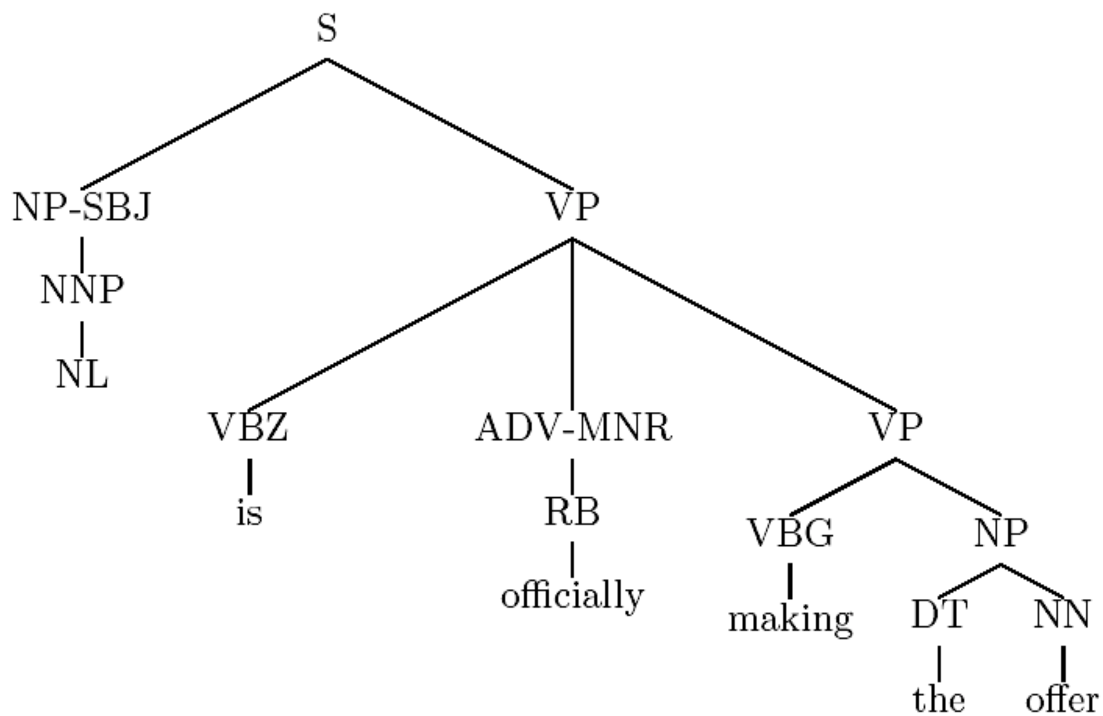
48 preterminals (tags):

- 36 POS tags, 12 other symbols (punctuation etc.)
- Simplified version of Brown tagset (87 tags)
  (cf. Lancaster-Oslo/Bergen (LOB) tag set: 126 tags)

14 nonterminals:

standard inventory (S, NP, VP,...)

A simple example

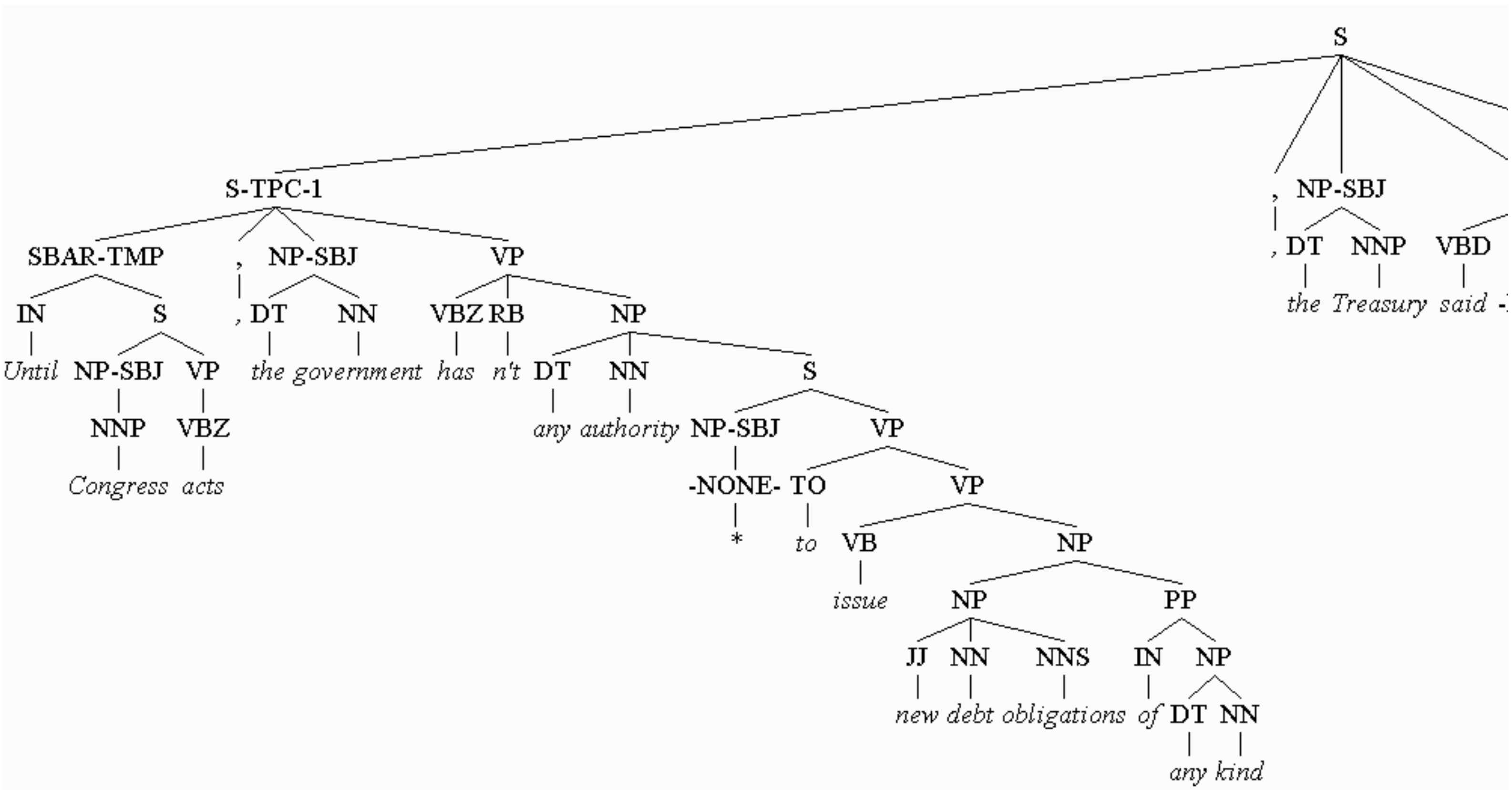


Relatively flat structures:
- There is no noun level
- VP arguments and adjuncts appear at the same level

Function tags, e.g. -SBJ (subject), -MNR (manner)

A more realistic (partial) example

Until Congress acts, the government hasn't any authority to issue new debt obligations of any kind, the Treasury said .... .



The Penn Treebank CFG

The Penn Treebank uses very flat rules, e.g.:

```
NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ `` SBAR '' NNS
```

– Many of these rules appear only once.
– Many of these rules are very similar.
– Can we pool these counts?

PCFGs in practice:

Charniak (1996) *Tree-bank grammars*

How well do PCFGs work on the Penn Treebank?

- Split Treebank into test set (30K words) and training set (300K words).
- Estimate a PCFG from training set.
- Parse test set (with correct POS tags).
- Evaluate unlabeled precision and recall

Sentence Lengths	Average Length	Precision	Recall
2-12	8.7	88.6	91.7
2-16	11.4	85.0	87.7
2-20	13.8	83.5	86.2
2-25	16.3	82.0	84.0
2-30	18.7	80.6	82.5
2-40	21.9	78.8	80.4

Two ways to improve performance

… change the (internal) grammar:

- **Parent annotation/state splits:**
  Not all NPs/VPs/DTs/… are the same.
  It matters where they are in the tree

… change the probability model:

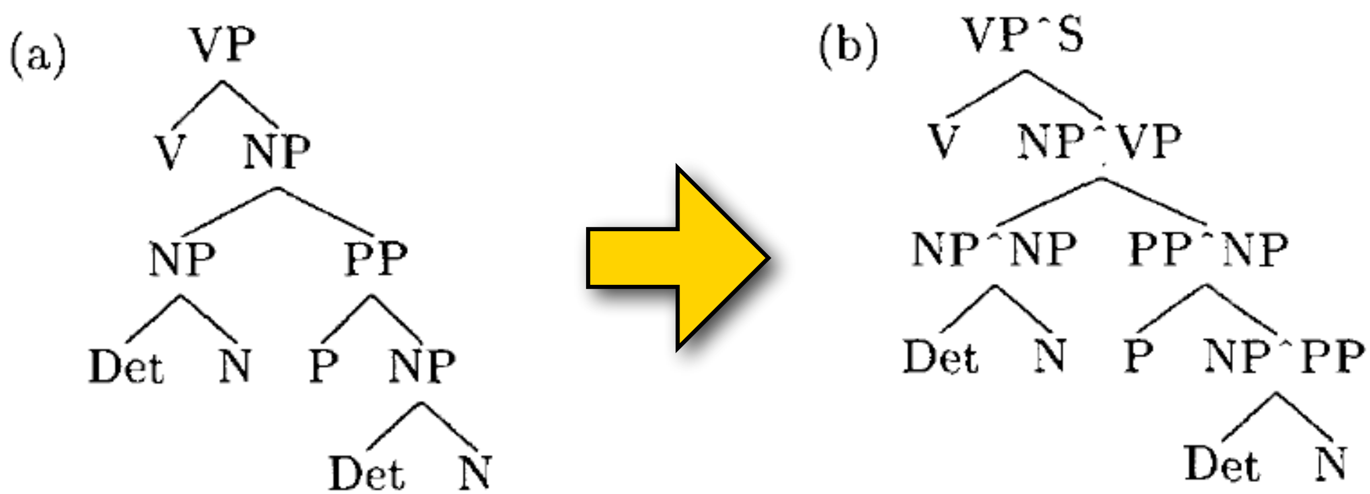
- **Lexicalization:**
  Words matter!
- **Markovization:**
  Generalizing the rules

The parent transformation

PCFGs assume the expansion of any nonterminal is independent of its parent.

- But this is not true: NP subjects more likely to be modified than objects.

We can change the grammar by adding the name of the parent node to each nonterminal



Markov PCFGs (Collins parser)

The RHS of each CFG rule consists of:
one head H_X, *n* left sisters L_i and *m* right sisters R_i:

$$X \longrightarrow \underbrace{L_n \ldots L_1}_{\text{left sisters}} \; H_X \; \underbrace{R_1 \ldots R_m}_{\text{right sisters}}$$

Replace rule probabilities with a generative process:
For each nonterminal X

- generate its head H_X (nonterminal or terminal)
- then generate its left sisters $L_{1..n}$ and a STOP symbol conditioned on H_X
- then generate its right sisters $R_{1...n}$ and a STOP symbol conditioned on H_X

Lexicalization

PCFGs can't distinguish between
"eat sushi with chopsticks" and *"eat sushi with tuna"*.

We need to take words into account!

$P(VP_{\text{eat}} \rightarrow VP\ PP_{\text{with chopsticks}} \mid VP_{\text{eat}})$

vs. $P(VP_{\text{eat}} \rightarrow VP\ PP_{\text{with tuna}} \mid VP_{\text{eat}})$

Problem: sparse data ($PP_{\text{with fatty|white|... tuna....}}$)
Solution: only take **head words** into account!

Assumption: each constituent has one head word.

Lexicalized PCFGs

At the root (start symbol S), generate the head word of the sentence, w_S, with $P(w_S)$

Lexicalized rule probabilities:
Every nonterminal is lexicalized: X_{w_x}
Condition rules $X_{w_x} \rightarrow \alpha Y \beta$ on the lexicalized LHS X_{w_x}
$P(X_{w_x} \rightarrow \alpha Y \beta \mid X_{w_x})$

Word-word dependencies:
For each nonterminal Y in RHS of a rule $X_{w_x} \rightarrow \alpha Y \beta$, condition w_Y (the head word of Y) on X and w_X:
$P(w_Y \mid Y, X, w_X)$

Dealing with unknown words

A lexicalized PCFG assigns zero probability
to any word that does not appear in the training data.

Solution:

Training: Replace rare words in training data
with a token 'UNKNOWN'.

Testing: Replace unseen words with 'UNKNOWN'

Refining the set of categories

Unlexicalized Parsing (Klein & Manning '03)

Unlexicalized PCFGs with various transformations of the training data and the model, e.g.:

– Parent annotation (of terminals and nonterminals):
distinguish preposition IN from subordinating conjunction IN etc.

– Add head tag to nonterminals
(e.g. distinguish finite from infinite VPs)

– Add distance features

Accuracy: 86.3 Precision and 85.1 Recall

The Berkeley parser (Petrov et al. '06, '07)

Automatically learns refinements of the nonterminals

Accuracy: 90.2 Precision, 89.9 Recall

Summary

The Penn Treebank has a large number of very flat rules.
Accurate parsing requires modifications to the basic PCFG model: refining the nonterminals, relaxing the independence assumptions by including grandparent information, modeling word-word dependencies, etc.

How much of this transfers to other treebanks or languages?