PROBABILISTIC PARSING

Stephen.Pulman@comlab.ox.ac.uk

- Probabilistic parsing: PFCG
- Heads and Complements
- Lexicalised CFGs
- Markovised Grammars
- Generative vs. Discriminative models
- the Averaged Perceptron classifier

Probabilistic Parsing

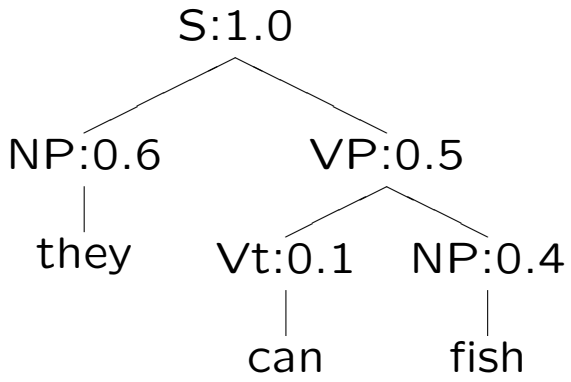
If we have a CFG grammar (or a treebank) we can associate probabilities with the rules. We want to find the tree that maximises P(Tree | Sentence), which here will be the same as P(Tree,Sentence).

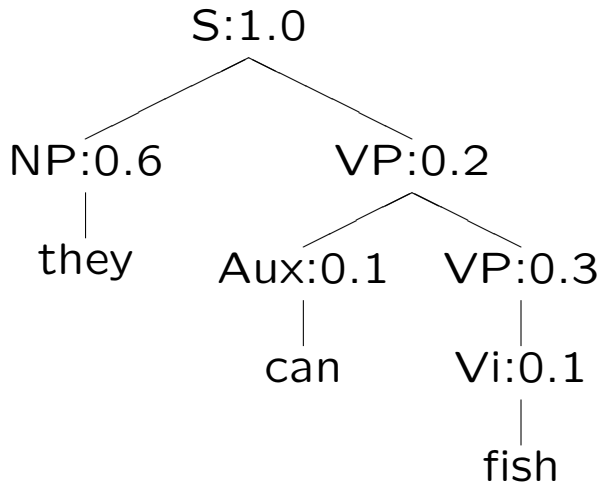
S	$\rightarrow$ NP VP	1.0	
NP	$\rightarrow$ they	0.6	
NP	$\rightarrow$ fish	0.4	
VP	$\rightarrow$ Aux VP	0.2	
VP	$\rightarrow$ Vi	0.3	
VP	$\rightarrow$ Vt NP	0.5	
Aux	$\rightarrow$ can	0.1	(etc)
Vi	$\rightarrow$ fish	0.1	(etc)
Vt	$\rightarrow$ can	0.1	(etc)

The probability of a derivation can be obtained by multiplying the probabilities of the rules figuring in it:

$$\text{P(Tree)} = \prod_i \text{P(Rule}_i\text{)}$$

If you have a parsed corpus, P(A $\rightarrow$ B) = n(A $\rightarrow$ B)/n(A $\rightarrow$ anything).





P(Tree1) = 1.0*0.6*0.5*0.1*0.4 = 0.012	P(Tree2)= 1.0*0.6*0.2*0.1*0.3*0.1 = 0.00036

This doesn't work well because it does not take account of the words used, or the context in which a rule is applied: e.g. NP subjects have a different distribution from NP objects.

Some necessary linguistic concepts

The **head** of a constituent is intuitively the most important member of it:

Constituent	Head
NP	N
VP	V
PP	P
Adjp	Adj
AdvP	Adv
S	VP
S with fronted X	fronted X

Conjunctions do not fit neatly...

The **complements** of a V, P, N, or Adj are the constituents required to make a complete phrase: nb every verb needs a subject.

put [the book] [on the shelf]	... the rumour [that he escaped]
* put the book _	... into [the lake]
* put _ on the shelf	... eager [to leave]

Other constituents are 'adjuncts' or optional modifiers.

WH-MOVEMENT

Questions:

Who put the book on the shelf
What did you put _ on the shelf?
What did you put the book on _ ?
On which shelf did you put the book _?
I wonder what you put _ on the shelf

Topicalisation

That book, I have never read _ carefully.
Afraid of the dark, she most certainly is not _.
On Friday, I have to go to London _.

Relative clauses:

The person that/who put the book on the shelf...
The shelf (that/which) you put the book on _ ...
The book (that/which) you put _ on the shelf ...

Context-free grammar analysis of movement

Begin with a simple CFG:

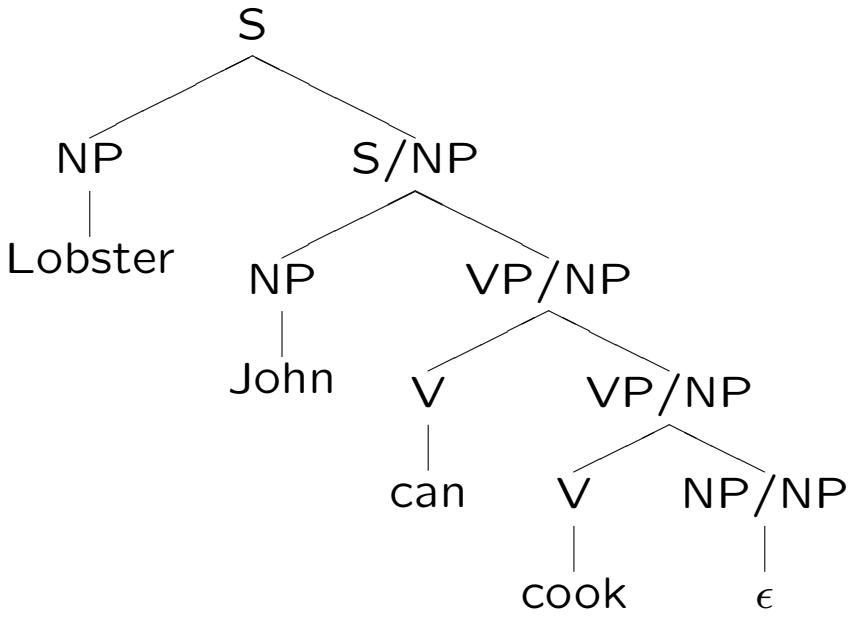
1. S $\rightarrow$ NP VP
2. VP $\rightarrow$ V S
3. VP $\rightarrow$ V NP
4. PP $\rightarrow$ P NP

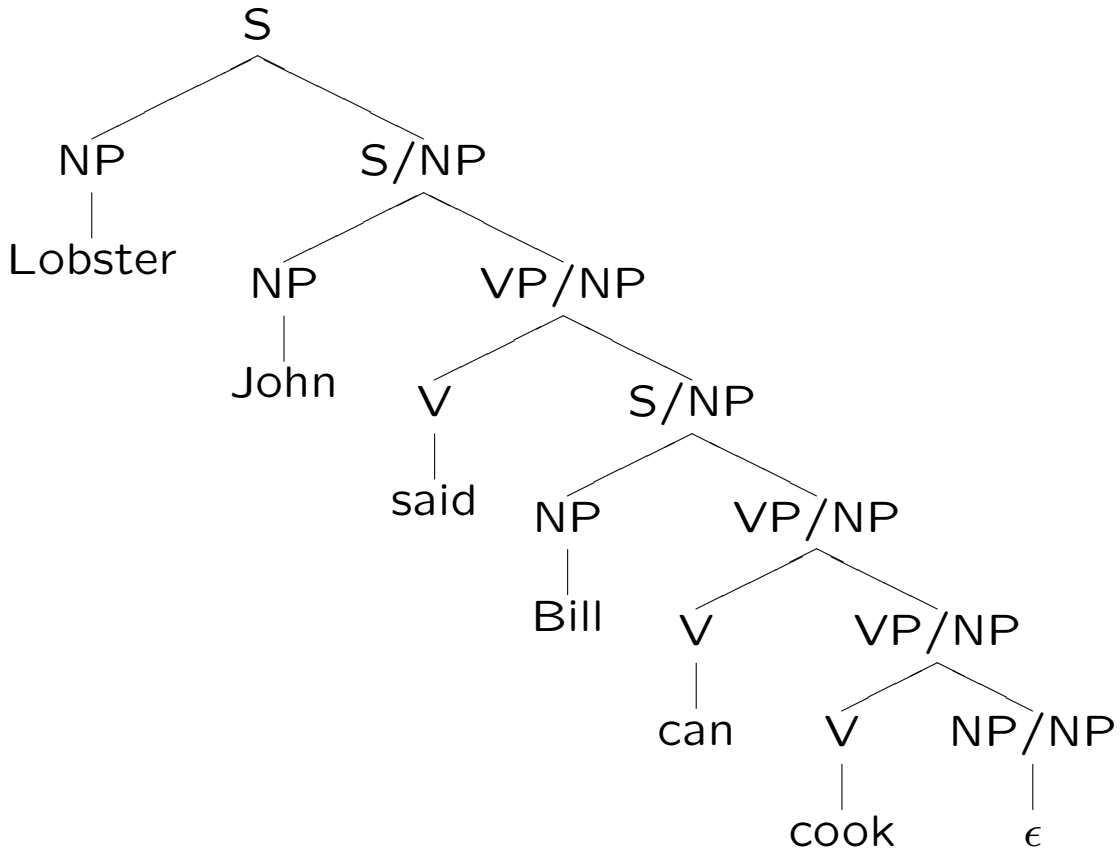
etc.

Now add rules for introducing 'slashed categories' (rule 5), rules for introducing 'gaps' (10) and rules for transmitting the slashed categories down the tree (6, 7, 8)

5. S $\rightarrow$ NP S/NP
6. S/NP $\rightarrow$ NP VP/NP
7. VP/NP $\rightarrow$ V S/NP
8. VP/NP $\rightarrow$ V NP/NP
9. PP/NP $\rightarrow$ P NP/NP
10. NP/NP $\rightarrow$ ϵ (i.e. the empty string)

etc.


S
NP
S/NP
Lobster
NP
VP/NP
John
V
VP/NP
can
V
NP/NP
cook
ϵ


S
NP
S/NP
Lobster
NP
VP/NP
John
V
S/NP
said
NP
VP/NP
Bill
V
VP/NP
can
V
NP/NP
cook
ϵ

The Penn Treebank

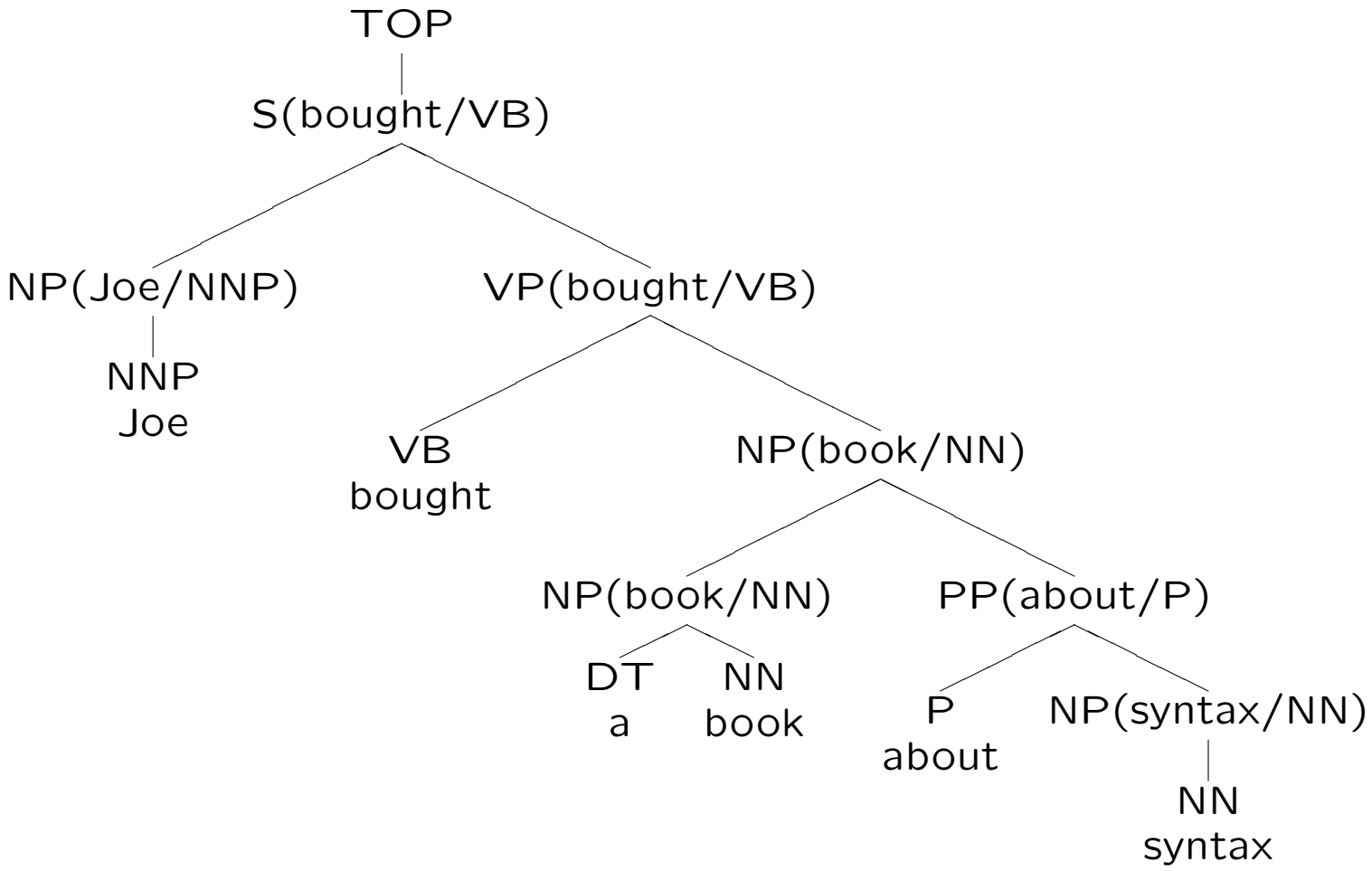
c. 1m words of tagged and annotated sentences:

```
(TOP
     (S
        (NP-SBJ
                (NP (NNP Pierre) (NNP Vinken)))
        (VP (MD will)
            (VP (VB join)
                 (NP (DET the) (NN board))
                 (PP-CLR (PREP as)
                        (NP (DET a) (NN director)))
                 (NP-TMP (NNP Nov.) (CD 29))
            )
         )
      )
 )
```

It is easy to induce a (probabilistic) CFG from the the treebank.

Adding richer information to a CFG ('lexicalising')

Associate head word and pos tag with each non-terminal:



We can now have more fine grained probabilities, but suffer from the 'data sparseness' problem, so have to approximate.

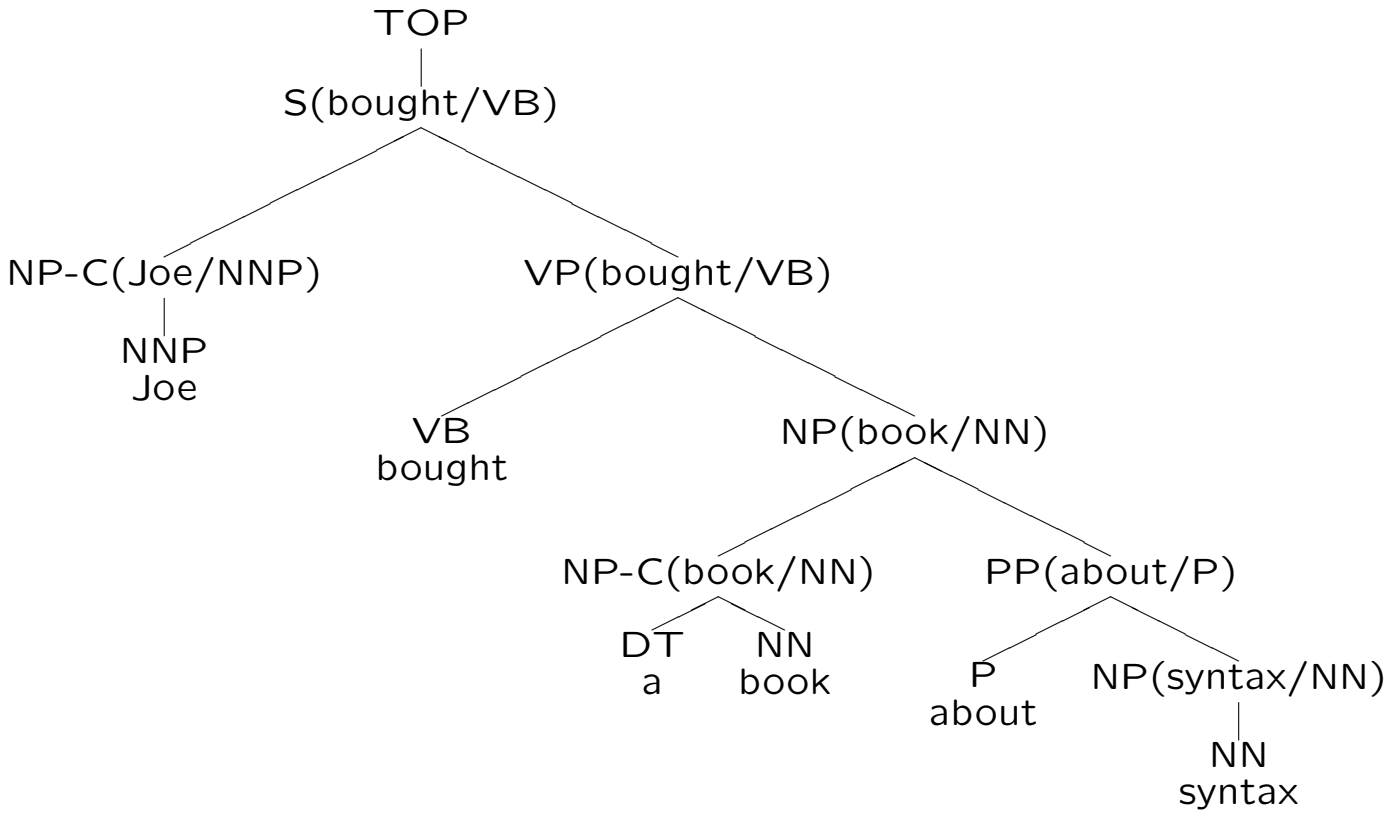
P(Mother $\rightarrow$ Left$_{m\ldots1}$ HeadCat/HeadWord Right$_{1\ldots n}$) =
P$_H$(HeadCat | Mother, HeadWord) *
P$_R$(Right$_{1\ldots n}$ | Mother, HeadCat, HeadWord) *
P$_L$(Left$_{m\ldots1}$ | Mother, HeadCat, HeadWord)

where P(Left$_{m\ldots1}$| X) = $\prod_{i=m\ldots1}$P(Left$_i$ | X). The Right case is analogous. We insert dummy 'STOP' symbols at the beginning and the end:
MOTHER $\rightarrow$ STOP LEFT HEAD RIGHT STOP
to keep things tidy when LEFT or RIGHT are empty. e.g. P(S(bought) $\rightarrow$ NP(Joe) VP(bought)) =
P$_H$(VP | S, bought) *
P$_L$(NP(Joe/NN) | S, VP, bought) *
P$_R$(STOP — S, VP, bought)

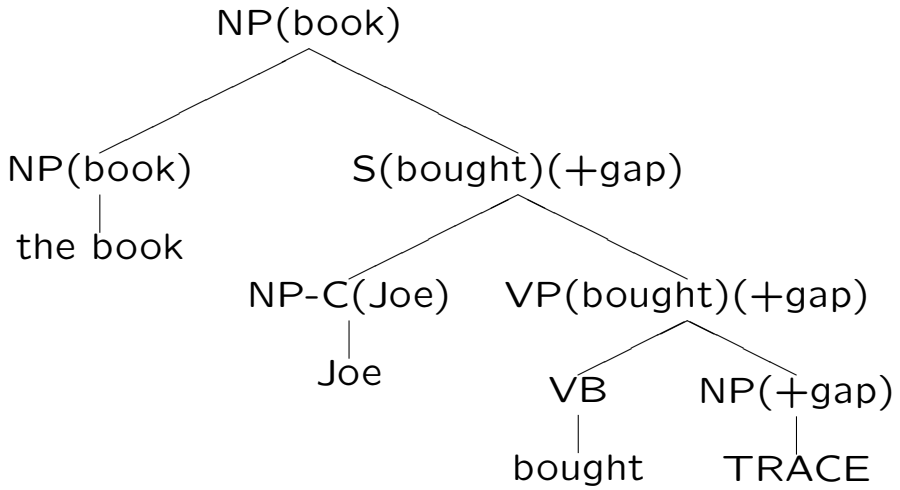
Collins also conditions the P$_L$ and P$_R$ probabilities on a distance measure, which takes into account the number of words between the current constituent and the head word, whether there are any punctuation symbols, and whether there is a verb intervening between the head and the current position.

Subcategorisation

We can also condition rules on subcategorisation requirements if the distinction between complements (-C) and adjuncts is marked in the corpus.



Wh-movement can be treated in a similar way by annotating categories as to whether they contain a ‘gap’.



'Backing off'

Typically we still encounter many cases not like anything in the training corpus. We have to 'back off' to a less fine-grained estimate:

e.g. P_H(HeadCat | Mother, HeadWord) can be approximated by
λ_1P(HeadCat | Mother, HeadTag) * λ_2P(Head | Mother)
where $\lambda_{1,2}$ are (trainable) smoothing parameters.

Training Procedure

Re-annotate the Penn Treebank sentences with -C and (gap +) information. This can be done by rule based pattern matching using the existing information.

Learn the rule probabilities according to the formulae above, using about 90% of the Treebank.

Parse the raw sentences from the remaining 10% and compare the most probable parse with the actual one in the treebank. (NB unknown word problem...)

Results (sentences up to 100 words long):

Labelled Precision =

$$\frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}}$$

Labelled Recall =

$$\frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}}$$

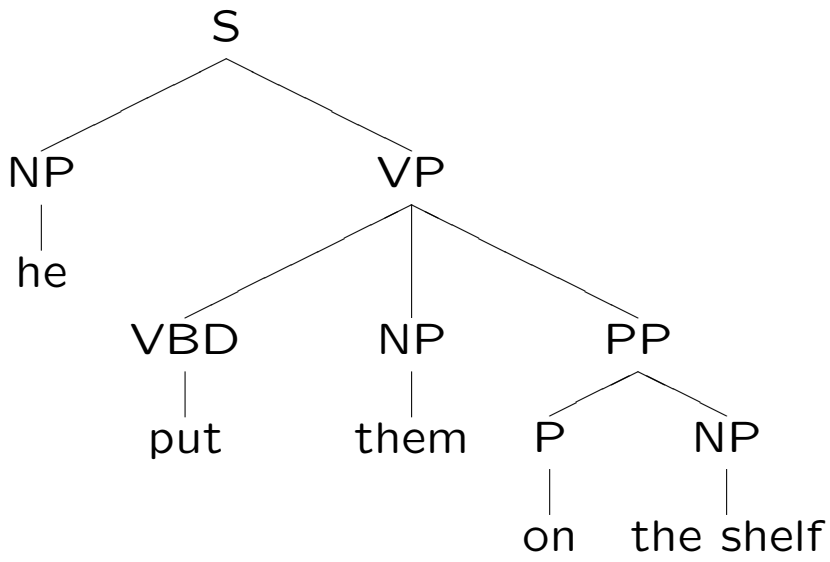
CB = Crossing Brackets = number of wrong constituent boundaries: e.g.

Treebank: `a [b][c d] e`
Proposed: `a [b c][d] e`

RESULTS	LR	LP	0 CBs
Simple	86.8	87.6	63.1
SubCat+Wh	87.5	88.1	63.9

Markovised Grammars

An alternative to lexicalisation is to include more context in grammar rules. This can be either horizontal, or vertical:



Vertical: (v=1)

S → NP↑S VP↑S
NP↑S → he
VP↑S → VBD↑VP NP↑VP PP↑VP
VBD↑VP → put
NP↑VP → them
etc.

Horizontal: (h=2)

VP → VBD NP PP:
replace VP in this context with ⟨VP:VBD+NP+PP⟩
⟨VP:VBD+NP+PP⟩ → ⟨VP:VBD+NP⟩ PP
⟨VP:VBD+NP⟩ → ⟨VP:VBD⟩ NP
⟨VP:VBD⟩ → VBD

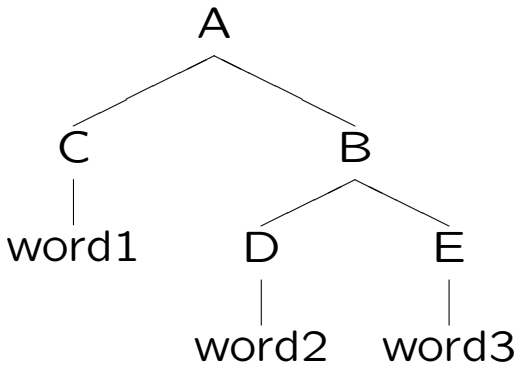
Using this method, along with further annotations encoding other vertical and horizontal context information, and some lexical info, Klein and Manning (2003) report an F-score of 87.04 on the Collins data, which is almost as good as Collins' original parser.

Generative vs. discriminative models

Models like Collins' or Charniak's are called 'generative', because you could use them to probabilistically regenerate the data they are trained on. They preserve information about the distribution of structures in the training data.

An alternative approach is to have some simpler model which has only information about possible structures, e.g. a CFG derived from the treebank. This could be used to find all possible parses. You can then use a separate model to filter, or to rank, the parses: this model can also be trained on a treebank (not necessarily the same one). Such a model 'discriminates' between parses. It is essentially a 'classifier' in the machine learning sense, that distinguishes between good and bad parses.

Classifiers When training a classifier, the training examples (and later, the unseen data) are typically represented as a set of 'features', which can be as rich as is desired, e.g.



'dominated by A', 'dominated by B', 'followed by D', 'preceded by D'...
'right sister is B','leftmost daughter is word2', etc.

These can also be combined to give more complex features.

Now train a classifier to separate good from bad parses. In methods like maximum entropy (log-linear) models, or voted or averaged perceptrons, each feature gets a weight corresponding to how well it predicts a good parse. The weights are adjusted iteratively in the training procedure until the optimal set of weights is obtained.

The averaged perceptron:

Training examples: (x_i,y_i), where x is an example, say, a sentence, and y is a label (e.g. the correct tree for that sentence)

Initial weight vector: $w = \langle 0,0,0,.... \rangle$

A function ϕ which maps an example (x,y) to a feature vector $\phi(x, y) \in \mathbb{R}^d$, where there are d possible features. The element are typically counts of the number of times the feature appears in the labelled example.

A function **G** which generates a set of possible labels **G**(x) for an example x.

Training: if we have n training examples and we iterate over them T times:

For t = 1...T, i = 1...n
    Calculate $z_i = argmax_{z \in \mathbf{G}(x_i)} \phi(x_i, z) \cdot w$

('$\cdot$' means 'inner' or 'dot' product: $\Sigma_{i=1}^{n} x_i y_i$)

If $z_i \neq y_i$ then $w = w + \phi(x_i, y_i) - \phi(x_i, z_i)$

Output: final w (simple case) or the average of all the w built during training.

For classification of new examples use $argmax_{z \in \mathbf{G}(x_i)} \phi(x_i, z) \cdot w$ to find the best ranked label.

The Stanford Parser:

`http://nlp.stanford.edu:8080/parser/`

The Clark and Curran parser:

`http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo`

Charniak, Eugene 2000 A Maximum-Entropy-Inspired Parser, Proceedings of NAACL 2000.

Collins, Michael 1997 Three Generative, Lexicalised Models for Statistical Parsing, Proceedings of the 35th meeting of ACL, Madrid, 16-23.

Michael Collins and Brian Roark. Incremental parsing with the Perceptron algorithm. ACL 2004.

Stephen Clark and James R. Curran, Parsing the WSJ using CCG and Log-Linear Models, ACL 2004

Clark, Stephen and Curran, James, 2007, Perceptron Training for a Wide-Coverage Lexicalized-Grammar Parser, ACL 2007 Workshop on Deep Linguistic Processing, Prague, Czech Republic, http://www.aclweb.org/anthology/W/W07/W 1202

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. ACL 2003, pp. 423-430.

David M. Magerman, Mitchell P. Marcus: Pearl: A Probabilistic Chart Parser. EACL 1991: 15-20

Jurafsky and Martin Chapter 12 (First Ed) Chapter 14 (Second Edition)

Manning, C. and Schütze, H. 1999 Foundations of Statistical Natural Language Processing, Cambridge Mass: MIT Press.