CS 4650/7650, Lecture 16: Modern Parsing Algorithms (The Parsing Wars)

Jacob Eisenstein

October 17, 2013

1 Review

PCFG parsing obtains the most likely derivation of a sentence.

- Production probabilities are read from a treebank.
- Bottom-up CKY is typically used for decoding.
- PCFG parsing with typical (Penn Treebank) non-terminals such as NP and VP doesn't work well.
- 75% test set accuracy when **training on the test set!**

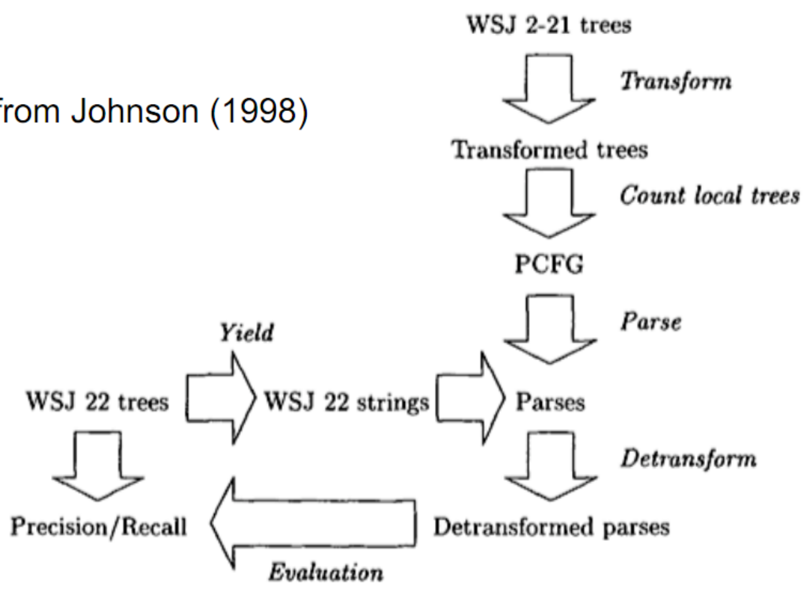
Last time, we talked about dependency parsing, which is less ambitious: a single dependency graph may correspond to multiple CFG derivations.

Today, we will focus on the innovations that improved constituent parsing from 72% to 91% accuracy.

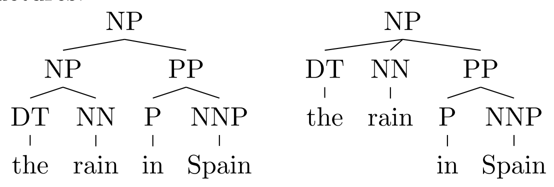
2 Tree transformations

Johnson proposed a series of transformations to PTB trees that improve parsing accuracy.

from Johnson (1998)



Flattening Johnson proposes "flattening" nested NPs to be more like VP structures.

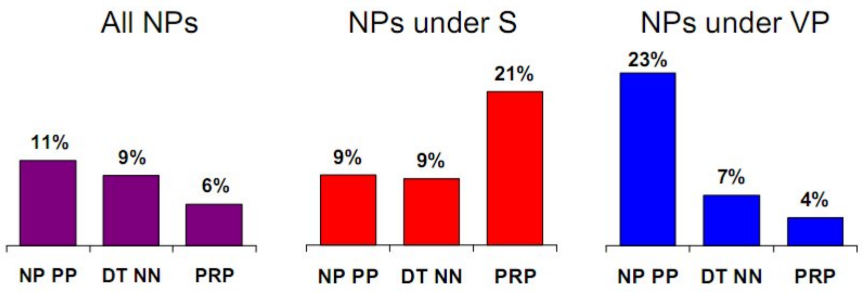


will this increase or decrease the size of the grammar?

Flattened rules are of course still context-free, but by reducing recursion, they allow more specific probabilities to be learned. This can eliminate the problems with rule subsumption that we saw earlier.

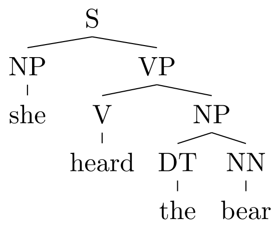
Think about the most extreme version of flattening. What would that look like?

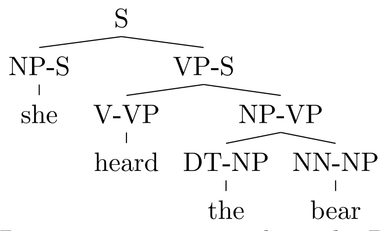
Parent annotation The expansion of an NP is highly dependent on its parent.



$$P(NP \to NP\ PP) = 11\%$$
$$P(NP(\text{under } S) \to NP\ PP) = 9\%$$
$$P(NP(\text{under } VP) \to NP\ PP) = 23\%$$

Parent annotation: augment each non-terminal with its parent.





Parent annotation weakens the PCFG independence assumptions

- which could help accuracy by allowing more fine-grained distinctions
- or could hurt accuracy because of data sparseness

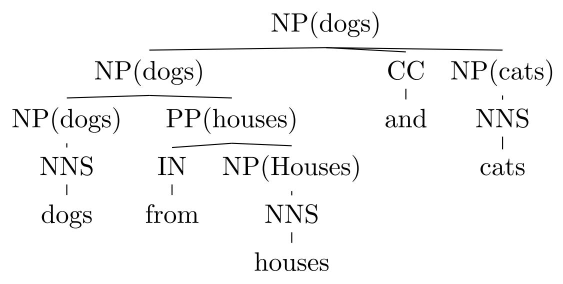
Overall, these transformations improve performance:

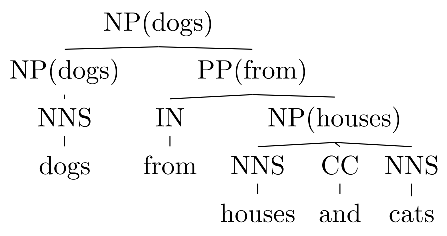
- Standard PCFG: 72% F-measure, 14,962 rules
- Parent-annotated PCFG: 80% F-measure, 22,773 rules
- In principle, parent annotation could have increased the grammar size much more drammatically, but many possible productions never occur, or are subsumed.

3 Lexicalization

A simple way to capture semantics is through the words themselves. We can annotate each non-terminal with **head** word of the phrase.

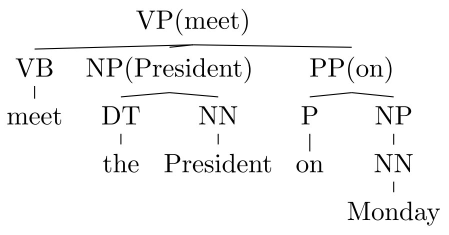
3.1 Example: coordination scope

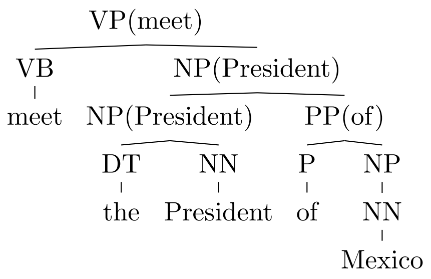




If $P(NP \rightarrow NP(\text{dogs})\ CC\ NP(\text{cats})) >$ $(NP \rightarrow NP(\text{houses})\ CC\ NP(\text{cats}))$, we should get the right parse.

3.2 Example: PP attachment





- $P(VP(meet) \rightarrow \alpha\ PP(on)) \gg P(NP(President) \rightarrow \beta\ PP(on)$
- $P(VP(meet) \rightarrow \alpha\ PP(of)) \ll P(NP(President) \rightarrow \beta\ PP(of)$
- In plain English:
  - *Meeting* happens *on* things.
  - *Presidents* are *of* things.

3.3 Lexicalization was a major breakthrough

Vanilla PCFG	72%
Head-annotated PCFG (Johnson 1998)	80%
Lexicalized PCFG (Collins 1997, Charniak 1997)	87-89%

Eugene Charniak (2000): "To do better, it is necessary to condition probabilities on the actual words of the sentence. This makes the probabilities much tighter."

One more example, subcategorization frames:

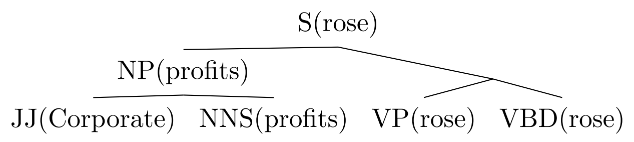
$$P(VP \to V\ NP\ NP) = 0.00151$$
$$P(VP(\text{said}) \to V(\text{said})\ NP\ NP) = 0.00001$$
$$P(VP(\text{gave}) \to V(\text{gave})\ NP\ NP) = 0.01980$$

3.4 How to do it

- Naively: just augment the non-terminals to include the cross-product of all PTB non-terminals and all words.
- This will never work
  - Number of possible productions: $\mathcal{O}(N^3V^3)$, $V \approx 10^5$
  - Too slow, too sparse (total amount of PTB data = 10^6)
- Two practical algorithms: Charniak (1997) and Collins (1999).

3.5 The Charniak Parser

The Charniak (1997) parser gives a relatively straightforward way to lexicalize PCFGs.



- Head probabilities capture "bilexical" phenomena, like the PP attachment (*President of Mexico*) example.
- Compute the head probability: $P(s_i|t_i, s_{p(i)}, t_{p(i)})$.
  - s_i is the head word of constituent i
  - $c(s_i)$ is the cluster of the head word of constituent i
  - t_i is the syntactic category
  - $p(i)$ is the parent of node i

- In the example, we have

$$
\begin{aligned}
&P(\textit{profits}|t_i = \text{NP}, t_{p(i)} = \text{S}, s_{p(i)} = \textit{rose}) \\
&= \frac{\text{count}(s_i = \textit{profits}, t_i = \text{NP}, t_{p(i)} = \text{S}, s_{p(i)} = \textit{rose})}{\text{count}(t_i = \text{NP}, t_{p(i)} = \text{S}, s_{p(i)} = \textit{rose})}
\end{aligned}
$$

- We multiply the head probability by a rule probability,

$$
\begin{aligned}
&P(\text{NP} \rightarrow \text{JJ NNS}|s_{p(i)} = \textit{rose}, t_i = \text{NP}, t_{p(i)} = \text{S}) \\
&= \frac{\text{count}(\text{NP} \rightarrow \text{JJ NNS}, t_i = \text{NP}, t_{p(i)} = \text{S}, s_{p(i)} = \textit{rose})}{\text{count}(t_i = \text{NP}, t_{p(i)} = \text{S}, s_{p(i)} = \textit{rose})}
\end{aligned}
$$

- This gives the values in the CKY table.

The rule probabilities capture phenomena like verb complement frames.

Local Tree	*come*	*take*	*think*	*want*
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

3.5.1 Data sparseness

The Penn Treebank is still the main dataset for syntactic analysis of English. Yet 1M words is not nearly enough data to accurately estimate lexicalized models.

- 965K constituents
- 66 examples of WHADJP
- only 6 of these aren't *how much* or *how many*

In the example above (*corporate profits rose*), the unsmoothed head probability is zero, as estimated from the PTB: there are zero counts of *profits* headed by *rose* in the treebank (hard to believe, but that's what Charniak says).

In general, bilexical counts are going to be very sparse. But the "backed-off" probabilities give a reasonable approximation. We will interpolate between them.

3.5.2 Smoothing the Charniak Parser

Head probability:

$$\begin{aligned}\hat{P}(s_i|t_i, s_{p(i)}, t_{p(i)}) = &\lambda_1 P_{mle}(s_i|t_i, s_{p(i)}, t_{p(i)}) \\ &+ \lambda_2 P_{mle}(s_i|t_i, \text{cluster}(s_{p(i)}), t_{p(i)}) \\ &+ \lambda_3 P_{mle}(s_i|t_i, t_{p(i)}) \\ &+ \lambda_4 P_{mle}(s_i|t_i)\end{aligned}$$

For example:

	$P(\textit{profit}\|NP, \textit{rose}, S)$	$P(\textit{corp.}\|JJ, \textit{profit}, NP)$
$P(s_i\|t_i, s_{p(i)}, t_{p(i)})$	0	.245
$P(s_i\|t_i, c(s_{p(i)}), t_{p(i)})$	.0035	.015
$P(s_i\|t_i, t_{p(i)})$	.00063	.0053
$P(s_i\|t_i)$	.00056	.0042

We have to tune $\lambda_1 \ldots \lambda_4$, and an equivalent set of parameters for the rule probabilities.

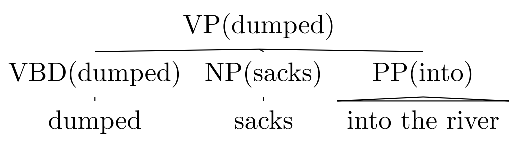
- The Charniak parser suffers from acute sparsity problems because it estimates the probability of entire rules.
- Another extreme would be to generate the children independently from each other.
  e.g., $P(S \rightarrow NP\ VP) \approx P_L(S \rightarrow NP) P_R(S \rightarrow VP)$
- Collins (1999) and Charniak (2000) go for a compromise, conditioning on the parent and the head child.

3.6 The Collins Parser

- The Charniak parser focuses on lexical relationships between children and parents.
- The Collins (1999) parser focuses on relationships between adjacent children of the same parent. It decomposes each rule as,

$$X \rightarrow L_i L_{i-1} \ldots L_1 H R_1 \ldots R_{j-1} R_j$$

- Each L and R is a child constituent of X, and they are generated from the head H outwards.
- The outermost elements of L and R are special • symbols.



To model this rule, we would compute:

$P(VP(dumped, VBD) \to \bullet\ VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, P)\ \bullet)$

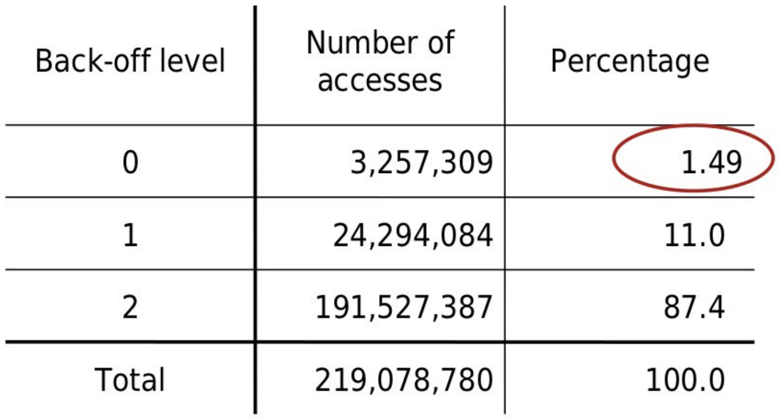
- Here's the generative process:
  - Generate the head: $P(H|LHS) = P(VBD(dumped, VBD)|VP(dumped, VBD))$
  - Generate the left dependent: $P_L(\bullet|VP(dumped, VBD), VBD(dumped, VBD))$
  - Generate the right dependent: $P_R(NP(sacks, NNS)|VP(dumped, VBD), VBD(dumped, VBD))$
  - Generate the right dependent: $P_R(NP(into, PP)|VP(dumped, VBD), VBD(dumped, VBD))$
  - Generate the right dependent: $P_R(\bullet|VP(dumped, VBD), VBD(dumped, VBD))$
- The rule probability is the product of these generative probabilities.
- Collins parser also conditions on a "distance" of each constituent from the head.

3.6.1 Smoothing the Collins Parser

- Estimation is eased by factoring the rule probabilities, but smoothing is still needed.

$$\begin{aligned}\hat{P}(R_i(rw_i, rt_i)|p(i), hw, ht) = &\lambda_1 P_{mle}(R_i(rw_i, rt_i)|p(i), hw, ht)\\ &+ \lambda_2 P_{mle}(R_i(rw_i, rt_i)|p(i), ht)\\ &+ \lambda_3 P_{mle}(R_i(rw_i, rt_i)|p(i))\end{aligned}$$

- We set λ using Witten-Bell smoothing.
- Is it worth modeling bilexical dependencies?



Back-off level	Number of accesses	Percentage
0	3,257,309	1.49
1	24,294,084	11.0
2	191,527,387	87.4
Total	219,078,780	100.0

- In general, bilexical probabilites are rarely available...
- ...but they are active in 29% of the rules in **top-scoring** parses.
- Still, they don't seem to play a big role in accuracy (Bikel 2004).

3.7 Summary of lexicalized parsing

- Lexicalized parsing resulted in substantial accuracy gains from our original PCFG:

Vanilla PCFG	72%
Parent-annotations	80%
Charniak (1997)	86%
Collins (1999)	87%

- But the explosion in the size of the grammar required elaborate smoothing techniques and made parsing slow.
- Treebank syntactic categories are too coarse, but lexicalized categories may be too fine.
- Is there a middle ground?

4 Accurate unlexicalized parsing

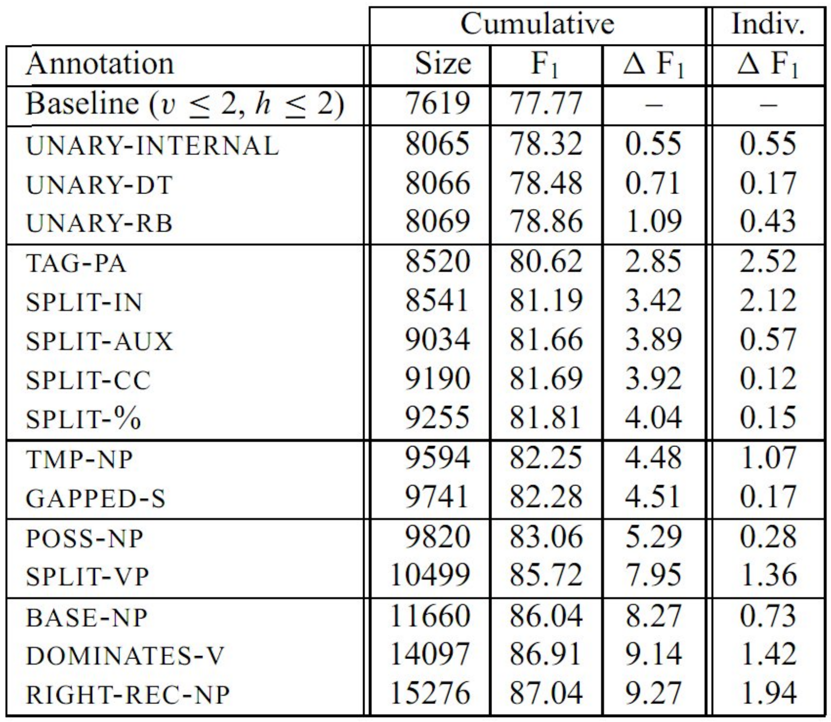
Klein and Manning (2003) revisit unlexicalized parsing, expanding on the ideas in (Johnson 1998).

The right level of linguistic detail is somewhere between treebank categories and individual words.

- For example, *on*/PP behaves differently from *of*/PP, but *cat*/N and *dog*/N do not.
- Approach: horizontal and vertical markovization, plus a series of linguistically-motivated splits to the Treebank categories.

4.1 Automatic state-splitting

Later work from Petrov and Klein (2007) automated the state-splitting process, using EM!



	Cumulative			Indiv.
Annotation	Size	F_1	Δ F_1	Δ F_1
Baseline ($v \leq 2, h \leq 2$)	7619	77.77	–	–
UNARY-INTERNAL	8065	78.32	0.55	0.55
UNARY-DT	8066	78.48	0.71	0.17
UNARY-RB	8069	78.86	1.09	0.43
TAG-PA	8520	80.62	2.85	2.52
SPLIT-IN	8541	81.19	3.42	2.12
SPLIT-AUX	9034	81.66	3.89	0.57
SPLIT-CC	9190	81.69	3.92	0.12
SPLIT-%	9255	81.81	4.04	0.15
TMP-NP	9594	82.25	4.48	1.07
GAPPED-S	9741	82.28	4.51	0.17
POSS-NP	9820	83.06	5.29	0.28
SPLIT-VP	10499	85.72	7.95	1.36
BASE-NP	11660	86.04	8.27	0.73
DOMINATES-V	14097	86.91	9.14	1.42
RIGHT-REC-NP	15276	87.04	9.27	1.94

- Assign a random subcategory to each node.
- Learn a PCFG.
- Apply the PCFG to relabel the nodes
  - subject to constraints of original annotations:
    VP3 can be relabeled as VP7, but not as an NP
- Repeat

They applied a split-merge heuristic to determine the number of subcategories for each phrase type. See slides for some more details.

5 Discriminative CFG parsing

- Generative parsers assume observations are conditionally independent given the label.
- This prohibits redundant features like morphology and word clusters
- This made a big difference in sequence labeling (25% error reduction).
- Can it help in CFG parsing?

5.1 Reranking

- Key idea: generate an N-best list of parses, learn a *ranking* function to score them (Collins, 2002)
- Advantage: can include arbitrary features.
- Can be as simple as perceptron
  - **Learning**
  $$w_n \leftarrow w_{n-1} + \eta(f(t,s) - f(\hat{t},s)) \tag{1}$$
  where $f(t,s)$ are the features of the correct parse and $f(\hat{t},s)$ are the features of the best-scoring parse.
  - **Decoding**: produce K parses from the generative model, return $\arg\max_k \boldsymbol{w}^{\mathsf{T}} f(t_k,s)$

Features

- **Conjoined** subtrees should have **similar depth and length**.
- English (and many other languages) have **right-branching** structure.
- Bigrams of horizontally neighboring tags
- Head-to-head dependencies, which can capture agreement
- "Heaviness" of non-terminals, and their proximity to the end of the sentence.

Vanilla PCFG	72%
Parent-annotations	80%
Lexicalized (Charniak 1997)	86%
Lexicalized (Collins 1999)	87%
Lexicalized (Charniak 2000)	90.1%
State-splitting (Petrov and Klein 2007)	90.6%
Reranking (Charniak and Johnson 2005)	91.0%

5.2 CRF parsers

Finkel et al (2008) propose a globally-normalized conditional models for parsing, the CFG-CRF model.

$$P(t|s;\theta) = \frac{1}{Z_s}\prod_{r\in t}\phi(r|s;\theta)$$
$$Z_s = \sum_{t'\in\tau(s)}\prod_{r\in t'}\phi(r|s;\theta)$$

- Model

- Each parse t is a collection of productions $\{r\}$.
- Each production has a non-negative potential $\phi(r|s;\theta) = e^{\boldsymbol{\theta}^\top \boldsymbol{f}(r,s)}$.
- The unnormalized score for a parse is the product of its potentials.
- The *partition function* Z_s is the sum of the scores for all possible parses for a sentence s.

- Features
  - If the features are local in t, we can use CKY to decode.
  - Features need not be local in s.
    (just like in discriminative sequence models)
  - standard PCFG stuff, with and without parent annotation
    (no need for complicated smoothing!)
  - lexicon features over words and tags
    (including prev word and next word, and unknown word classes)
  - bigrams and trigrams of *word classes* under each subtree
- **Decoding** is just like generative PCFG parsing
  - Multiply potentials ϕ rather than probabilities.
  - We need only the unnormalized score $\prod_{r \in t} \phi(r|s;\theta)$.
- **Learning** is just like logistic regression:

$$\mathcal{L} = \left[\sum_{(t,s)\in\mathcal{D}} \left(\sum_{r\in t} \sum_i \theta_i f_i(r,s) \right) - Z_s \right] + \sum_i \frac{\theta_i^2}{2\sigma^2}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \left[\sum_{(t,s)\in\mathcal{D}} \left(\sum_{r\in t} \sum_i f_i(r,s) \right) - E_\theta[f_i|s] \right] + \frac{\theta_i}{\sigma^2}$$

  - compute Z_s using the inside algorithm
  - compute $E_\theta[f_i|s]$ using inside-outside
  - But unfortunately, $\mathcal{O}(N^3 G)$ is still too slow, so tricks are needed.
    * Run a non-probabilistic CFG to prune away productions which do not lead to any valid parse.
    * **Parallelization** via stochastic gradient descent:

Final results table

Vanilla PCFG	72%
Parent-annotations	80%
Lexicalized (Charniak 1997)	86%
Lexicalized (Collins 1999)	87%
State-splitting (Petrov and Klein 2007)	90.6%
Reranking (Charniak and Johnson 2005)	91.0%
CFG-CRF (Finkel et al 2008)	89.0

- CRF-CFG is better than basic generative parsers, worse than reranking.
- The key advantage of this approach is the simplicity:
  no need for complicated smoothing or backoff, just throw redundant information at the learner and let it sort things out.
- An alternative CRF based on tree-adjoining grammar scored 91.1 (Carreras et al, 2008)

6 Summary of CFG parsing

- A big part of parsing research has been figuring out the right level of description:
  - Unlexicalized PCFGs on treebank categories are too coarse.
  - Lexicalized parsers start with very rich probability models, and then apply smoothing for tractability.
  - State-splitting approaches start with the treebank categories and split as needed.
- Reranking approaches can learn rich feature representations, but can only apply them to a limited set of possible parses.
- Globally-normalized conditional parsers learn feature-based models and apply them to decode over the entire space of possible parses.