CS 4650/7650, Lecture 13:
Parsing 2

Jacob Eisenstein

October 1, 2013

1 Recap

- Regular languages are a strict subset of context-free languages (CFLs)
- **Context-free grammars** (CFGs) define CFLs, just as **regular expressions** define RLs.
- **Parsing** is the task of determining whether a string can be **derived** from a CFG through a series of **productions**. If a string can be derived from a CFG, it is in the corresponding CFL.
- Productions go from non-terminals to other non-terminals or terminal symbols.
- In natural language, non-terminals correspond to **constituents**: sets of words that tend to behave like syntactic units.

2 A simple grammar of English

2.1 Noun phrases

Let's start with noun phrases:

- ***She*** *sleeps* (Pronoun)
- ***Arlo*** *sleeps* (Proper noun)
- ***Fish*** *sleep* (Mass noun)
- ***The fish*** *sleeps* (determiner + noun)
- ***The blue fish*** *sleeps* (DT + JJ + NN)
- ***The girl from Omaha*** *sleeps* (NP + PP)
- ***The student who ate 15 donuts*** *sleeps* (NP + RelClause)

So overall, we can summarize this fragment as

$$\begin{aligned} \text{NP} &\rightarrow \text{PRP} \mid \text{NNP} \mid \text{DT } \textsc{Nom} \mid \textsc{Nom} \\ \textsc{Nom} &\rightarrow \textsc{Adj}\text{P NN} \mid \text{NN} \mid \textsc{Nom}\text{ PP} \mid \textsc{Nom}\ \textsc{RelClause} \end{aligned}$$

We're leaving out some detail, like pluralization and possessives, but you get the idea.

2.2 Adjectival and prepositional phrases

- *Very funny*
- *The **large, blue** fish*
- *The man **from la mancha***

$$\begin{aligned} \textsc{Adj}\text{P} &\rightarrow \text{JJ} \mid \text{RB } \textsc{Adjp} \mid \text{JJ } \textsc{Adjp} \\ \text{PP} &\rightarrow \text{IN NP} \mid \text{TO NP} \end{aligned}$$

2.3 Verb phrases

- *She **sleeps***
- *She **sleeps restlessly***
- *She **sleeps at home***
- *She **eats sushi***[1]
- *She **gives John sushi***

$$\text{VP} \rightarrow \text{V} \mid \text{VP RB} \mid \text{VP PP} \mid \text{V NP} \mid \text{V NP NP} \mid \text{V NP RB}$$

But what about **She sleeps sushi* or **She speaks John Japanese*?

- Classes of verbs can take different numbers of arguments.
- This is called **subcategorization**

$$\begin{aligned} &\text{VP} \rightarrow \text{V-}\textsc{intrans} \mid \text{V-}\textsc{trans}\text{ NP} \mid \text{V-}\textsc{ditrans}\text{ NP NP} \\ &\text{VP} \rightarrow \text{VP RB} \mid \text{VP PP} \end{aligned}$$

We would also need to handle modal and auxiliary verbs that allow us to create complex tenses, like *She will have eaten sushi* but not **She will have eats sushi*.

[1] Sushi examples from Julia Hockenmaier

2.4 Sentences

- *She eats sushi*

$$S \rightarrow NP\ VP$$

- *Sometimes, she eats sushi*

$$S \rightarrow \textsc{AdvP}\ S$$

- *In Japan, she eats sushi*

$$S \rightarrow PP\ S$$

- What about **I eats sushi*, **She eat sushi*??

$$S \rightarrow NP.3S\ VP.3S \mid NP.N3S\ VP.N3S$$

In general, we need **features** to capture this kind of agreement.

2.5 Conjunctions

- *She eats sushi and candy*

$$NP \rightarrow NP\ \textit{and}\ NP$$

- *She eats sushi and drinks soda*

$$VP \rightarrow VP\ \textit{and}\ VP$$

- *She eats sushi and he drinks soda*

$$S \rightarrow S\ \textit{and}\ S$$

- *fresh and tasty sushi*

$$\textsc{AdjP} \rightarrow JJ\ \textit{and}\ JJ$$

We'd need a little more cleverness to properly cover groups larger than two.

2.6 Odds and ends

- *I gave sushi to the girl* ***who eats sushi***. This is a relative clause,

$$\textsc{RelClause} \rightarrow \textit{who}\ VP \mid \textit{that}\ VP$$

- *I took sushi from the man* ***offering sushi***. This is a gerundive postmodifier.

$$\textsc{Nom} \rightarrow \textsc{Nom}\ \textsc{Gerund}VP$$
$$\textsc{Gerund}VP \rightarrow VBZ \mid VBZ\ NP \mid VBZ\ PP \mid \ldots$$

- ***Can*** *she eat sushi?* (notice it's not *eats*)

$$S \rightarrow \textsc{Aux}\ NP\ VP$$

- ... and many more

3 Grammar design

Our goal is a grammar that avoids

- **Overgeneration**: deriving strings that are not grammatical.
- **Undergeneration**: failing to derive strings that are grammatical.

To avoid undergeneration, we would need thousands of productions.

Typically, grammars are defined in conjunction with large-scale **treebank** annotation projects.

- An annotation guideline specifies the non-terminals and how they go together.
- The annotators then apply these guidelines to data.
- The grammar rules can then be read off the data.

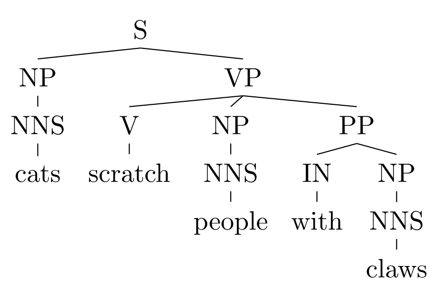
The Penn Treebank contains one million parsed words of Wall Street Journal text from the 1990s.

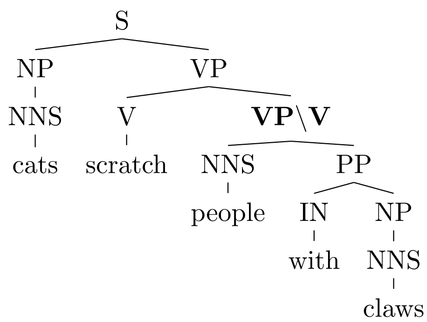
4 Grammar equivalence and normal form

- Grammars are weakly equivalent if they generate the same strings.
- Grammars are strongly equivalent if they generate the same strings **and** assign the same phrase structure to each sentence.
- In Chomsky Normal Form (CNF), all productions are either:

$$A \rightarrow BC$$
$$A \rightarrow a$$

  - All CFGs can be converted into a weakly equivalent grammar in CNF.
  - This is very handy for parsing algorithms.





- Binarization is easy:
  group right children into new non-terminals.
- Un-binarization is important!
  people with claws is not a constituent in the original parse.
- Unary productions are best handled by modifying the algorithm.

5 Parsing

Parsing is the process of determining whether a sentence is in a context-free language, by searching for a legal derivation.

Some possibilities:

- **Top-down**: start with the start symbol, and see if we can derive the sentence.
- **Bottom-up**: combine the observed symbols using whatever productions we can, until we reach the start symbol
- **Left-to-right**: move through the input, incrementally building a parse tree

Before we get into these different possibilities, let's see whether exhaustive search is possible. Suppose we only have one non-terminal, X, and it has binary productions

$$\text{X} \rightarrow \text{X X}$$
$$\text{X} \rightarrow \textit{the girl} \mid \textit{ate sushi} \mid \ldots$$

How many different ways could we parse a sentence? This is just equal to the number of binary bracketings of the words in the sentence, which is a Catalan number. Catalan numbers grow **super-exponentially** in the length of the sentence, $C_n = \frac{(2n)!}{(n+1)!n!}$.

5.1 Why parsing?

Lease *et al.*(AAAI 2006) identify several applications:

- **Language modeling**. A probabilistic parsing model can assign a likelihood to sentences that captures their grammaticality over long-range dependencies — unlike n-grams. This is relevant for speech recognition and machine translation.
- **Information extraction**
  - Entities are typically realized as NP constituents (e.g., *Barack Obama*, *The President of the French Football Association*)
  - Relations and their arguments can be detected from VPs and their arguments. (e.g., *We eat sushi and tempura*)
- **Question answering** can be performed by matching queries (*Who is the hero of Lord of the Rings*) and to candidate answers (*The hero of Lord of the Rings is Frodo*, *Frodo is the hero of Lord of the Rings*, *Lord of the Rings has a hero named Frodo*)
- **Machine translation** involves reorderings which should be easier given a hierarchical syntactic representation.

6 CKY parsing

CKY is a bottom-up parsing allows us to test whether a sentence is in a context-free language, without considering all possible parses. First we form small constituents, then try to merge them into larger constituents.

Let's start with an example grammar:

$$\begin{aligned}
S \rightarrow & VP\ NP \\
NP \rightarrow & NP\ PP \mid \textit{we} \mid \textit{sushi} \mid \textit{chopsticks} \\
PP \rightarrow & P\ NP \\
P \rightarrow & \textit{with} \\
VP \rightarrow & VP\ NP \mid VP\ PP \mid \textit{eat}
\end{aligned}$$

Suppose we encounter the sentence *We eat sushi with chopsticks.*

- The first thing that we notice is that we can apply unary productions to obtain NP VP NP P NP
- Next, we can apply a binary production to merge the first NP VP into an S.
- Or we could merge VP NP into VP
- ... and so on

Let's systematize this. Here is the CKY algorithm:

for j : [1,N] **do**
  $t[j-1,j] \leftarrow \{A|A \rightarrow x_j \in R\}$
  for i : [j-2, 0] **do**
    for k : [i+1, j-1] **do**
      $t[i,j] \leftarrow t[i,j] \cup \{A|A \rightarrow BC \in R, B \in t[i,k], C \in t[k,j]\}$
    end for
  end for
end for

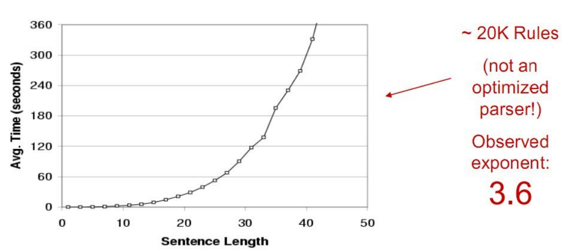
- If $A \in t[n,m]$, this means that the span $\boldsymbol{x}_{n:m-1}$ can be derived from non-terminal A.
- If $S \in t[0,N]$, this means that the entire string $\boldsymbol{x}$ can be derived from the start symbol S, so the string is in the language.

To handle unary transitions, we compute the *unary closure* of each non-terminal.

- e.g., if $S \rightarrow VP$, $VP \rightarrow V$, then add $S \rightarrow V$
- At each table entry $t[i,j]$
  - For each non-terminal $A \in t[i,j]$
    * Add all elements of the reflexive unary closure for A
- e.g., $\{eat, \mathrm{V}, \mathrm{VP}, \mathrm{S}\}$

Complexity What is the complexity of CKY?

- Space compexity: $\mathcal{O}(L^2|N|)$
- Time complexity: $\mathcal{O}(N^3|R|)$
- L is length of sentence,
  $|N|$ is the number of non-terminals,
  $|R|$ is the number of production rules
- But in practice...



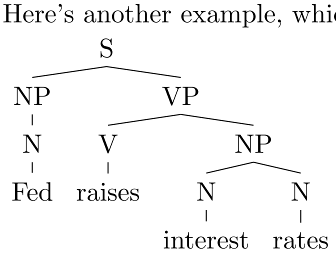
It's worse than worst-case! (figure from Dan Klein)

- Longer sentences "unlock" more of the grammar.

7 Ambiguity in parsing

- Syntactic ambiguity is endemic to natural language:[2]
  - Attachment ambiguity: *we eat sushi with chopsticks*, *I shot an elephant in my pajamas.*
  - Modifier scope: *southern food store*
  - Particle versus preposition: *The puppy tore up the staircase.*
  - Complement structure: *The tourists objected to the guide that they couldn't hear.*
  - Coordination scope: *"I see," said the blind man, as he picked up the hammer and saw.*
  - Multiple gap constructions: *The chicken is ready to eat*
- In morphology, we didn't just want to know which derivational forms are *legal*, we wanted to know which were likely.
- Syntactic parsing is all about choosing among the many, many legal parses for a given sentence.

Here's another example, which we've seen before:



- A minimal grammar permits 36 parses!
- Real-size broad coverage grammars permit millions of parses.

Classical parsers faced a tradeoff:

- broad coverage with tons of ambiguity...
- or limited coverage in exchange for constraints on ambiguity

Consequently, deterministic parsers produced no analysis for many sentences.

[2]Examples borrowed from Dan Klein

7.1 Local solutions

Some ambiguity can be resolved locally:

- [*imposed* [*a ban* [*on asbestos*]]]
- [*imposed* [*a ban*][*on asbestos*]]
- Hindle and Rooth (1990) proposed a likelihood ratio test:

$$LR(v, n, p) = \frac{P(p|v)}{P(p|n)} = \frac{P(\textit{on}|\textit{imposed})}{P(\textit{on}|\textit{ban})}$$

  where we select VERB attachment if $LR(v, n, p) > 1$.

- But the likelihood-ratio approach ignores important information, like the phrase being attached.
  - ...[*it* [*would end* [*its venture* [*with Maserati*]]]]
  - ...[*it* [*would end* [*its venture*][*with Maserati*]]]
- The likelihood ratio gets this wrong
  - $P(\textit{with}|\textit{end}) = \frac{607}{5156} = 0.118$
  - $P(\textit{with}|\textit{venture}) = \frac{155}{1442} = 0.107$

  Other features (e.g., *Maserati*) argue for noun attachment. How can we add them?

Machine learning solutions Ratnaparkhi et al (1994) propose a maximum-entropy (logistic regression) approach:

$$P(\mathrm{N}|\textit{would end its venture with Maserati}) = \frac{e^{\boldsymbol{w}^\top \boldsymbol{f}(\textit{would end its venture with Maserati})}}{1 + e^{\boldsymbol{w}^\top \boldsymbol{f}(\textit{would end its venture with Maserati})}}$$

Features include n-grams and word classes from hierarchical word clustering; accuracy is roughly 80%.

Collins and Brooks (1995) argued that attachment depends on four **heads**:

- the preposition (*with*)
- the VP attachment site (*end*)
- the NP attachment site (*venture*)
- the NP to be attached (*Maserati*)

They propose a backoff-based approach:

- First, look for counts of the tuple ⟨*with, Maserati, end, venture*⟩

- If none, try $\langle$*with, Maserati, end*$\rangle$ + $\langle$*with,end,venture*$\rangle$ + $\langle$*with,Maserati, venture*$\rangle$
- If none, try $\langle$*with, Maserati*$\rangle$ + $\langle$*with,end*$\rangle$ + $\langle$*with,venture*$\rangle$
- If none, try $\langle$*with*$\rangle$

Accuracy is roughly 84%. This approach of combining relative frequency estimation, smoothing, and backoff was very characteristic of late 1990s statistical NLP.

7.2 Beyond local solutions

Framing the problem as attachment ambiguity is limiting:

- assumes the parse is mostly done, leaving just a few attachment ambiguities to solve
- But realistic sentences have more than a few syntactic interpretations.
- Attachment decisions are interdependent:
  - *Cats scratch people with claws with knives.*
  - We may want to attach *with claws* to *scratch*.
  - But then we have nowhere to put *with knives*.

The task of statistical parsing is to produce a single analysis that resolves all syntactic ambiguities.

8 PCFGs

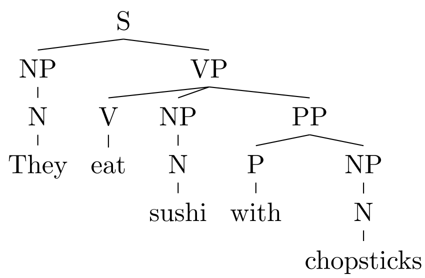
PCFGs extend the CFG by adding probability to each production:

S	$\rightarrow NP\ VP$	0.9
S	$\rightarrow S\ conj\ S$	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

The probabilities for all productions involving a single LHS must sum to 1:

$$\sum_{\alpha} P(X \rightarrow \alpha | X) = 1$$

- Let τ be the derivation of a string.
- The probability $P(\tau)$ is just the product of all the productions in the derivation.
- The **yield** of a parse tree is the string of terminal symbols that can be read off the leaf nodes.
- The set $\{\tau : S = \text{yield}(\tau)\}$ is exactly the set of all derivations of S in a CFG G.



In probabilistic parsing, we want the parse τ that maximizes $P(\tau|S)$.

$$\begin{aligned}
\arg\max_{\tau} P(\tau|S) =& \arg\max_{\tau} \frac{P(\tau, S)}{P(S)} \\
=& \arg\max_{\tau} P(\tau, S) \\
=& \arg\max_{\tau} P(S|\tau)P(\tau) \\
=& \arg\max_{\tau : S = \text{yield}(\tau)} P(\tau)
\end{aligned}$$

8.1 Estimation

Where do the probabilities come from?

- As in supervised HMMs, estimation is easy (for now!).
- PCFG probabilities can be estimated directly from a treebank:

$$P(VP \to VP\ PP) = \frac{\text{count(VP} \to \text{VP PP)}}{\text{count(VP)}}$$

- The Penn Treebank is 1M words of parse-annotated text, from which we can estimate these probabilities.

8.2 Three basic problems for PCFGs

Let $\tau \in T$ be a derivation, S be a sentence, and λ a PCFG.

- **Decoding**: Find $\hat{\tau} = \arg\max_{\tau} P(\tau, S; \lambda)$

- **Likelihood**: Find $P(w;\lambda) = \sum_{\tau} P(\tau, S; \lambda)$
- **(Unsupervised) Estimation**: Find $\arg\max_{\lambda} P(S_{1...N}|\lambda)$

	Sequences	Trees
model	HMM	PCFG
decoding	Viterbi algorithm	CKY
decoding complexity	$\mathcal{O}(N^2|K|)$	$\mathcal{O}(N^3|R|)$
likelihood	forward algorithm	inside algorithm
marginals	forward-backward	inside-outside

8.3 CKY with probabilities

for j : [1,N] **do**
  for X : $tags(s_j)$ **do**
    $t[X, j-1, j] \leftarrow P(X, s_j)$
  end for
  for i : [j-2, 0] **do**
    for $(X \to Y\ Z) \in R$ **do**
      for k : [i+1, j-1] **do**
        $t[X,i,j] \leftarrow t[X,i,j] \oplus (P(X \to Y\ Z) \otimes t[Y,i,k] \otimes t[Z,k+1,j])$
      end for
    end for
  end for
end for

In this algorithm,

- $t[A, m, n]$ is the score for generating the span $x_m : x_{n-1}$ from non-terminal A
- The recurrence $t[X,i,j] \leftarrow t[X,i,j] \oplus (P(X \to Y\ Z) \otimes t[Y,i,k] \otimes t[Z,k+1,j])$ combines
  - The previous score $t[X,i,j]$
  - The score of the production $P(X \to Y\ Z)$
  - The score of the left subtree $t[Y,i,k]$
  - The score of the right subtree $t[Z,k+1,j]$
- We iterate over all legal productions $(X \to Y\ Z) \in R$, for each midpoint $i < k < j$.

Let's try this on the *we eat sushi with chopsticks* example.

- **Boolean semiring**
  - $\oplus = \vee$, $\otimes = \wedge$, $\overline{0} = False$.
  - $t[X,n,m] = True$ iff there is some derivation from non-terminal X to the span $\boldsymbol{w}_{n:m}$.

- $t[S, 0, N] = True$ iff sentence $\boldsymbol{w}_{1:N}$ is in the language.
- This is equivalent to deterministic CKY.

- **Tropical semiring**
  - $\oplus = \max$, $\otimes = \times$, $\bar{0} = 0$.
  - $t[X, n, m] = \max_\tau P(X \rightarrow_\tau \boldsymbol{w}_{n:m})$, the probability of the best derivation of $\boldsymbol{w}_{n:m}$
  - $t[S, 0, N] = \max_\tau P(X \rightarrow_\tau \boldsymbol{w})$, the probability of the best parse. If we keep back pointers, we can recover it.

- **Probability semiring**
  - $\oplus = +$, $\otimes = \times$, $\bar{0} = 0$.
  - $t[X, n, m] = \sum_\tau P(X \rightarrow_\tau \boldsymbol{w}_{n:m})$, the total probability of all derivations of $\boldsymbol{w}_{n:m}$ from X.
  - $t[S, 0, N] = \sum_\tau P(S \rightarrow_\tau \boldsymbol{w}) = P(\boldsymbol{w})$, the probability of all derivations of $\boldsymbol{w}$.
  - This is the **inside algorithm**, similar to the **forward** algorithm from Hidden Markov Models.
  - Remember the **backward** algorithm? There is an equivalent **outside** algorithm.
  - Just as the forward-backward algorithm computes marginal probabilities for tags, the inside-outside algorithm computes marginal probabilities for non-terminals over spans. We'll talk about this after the midterm.

To handle unary transitions, we have to search the unary closure at each position, computing

$$t[X, n, m] = t[X, n, m] \oplus (P(X \rightarrow Y) \otimes t[Y, n, m]),$$

until we have tried all elements in the unary closure.

8.4 Evaluation

PARSEval is used to score parsing output, based on the number of correct **spans**.

- In **labeled** evaluation, the span and label must be correct.
- In **unlabeled** evaluation, only the span must be correct.

There is a recall/precision tradeoff

- **Recall**: how many of the true spans were predicted?

- **Precision**: how many of the predicted spans were true?
- **F-measure**: $F = \frac{2PR}{P+R}$

Can you see how to get perfect precision?

Let's try evaluating a sentence:

- Key: (She (eats (sushi) (with chopsticks)))
- Response: (She (eats (sushi (with chopsticks))))

Ground Truth

label	Text	Start	End
S	*she eats sushi with chopsticks*	1	6
NP	*she*	1	2
VP	*eats sushi with chopsticks*	2	6
V	*eats*	2	3
N	*sushi*	3	4
PP	*with chopsticks*	4	6
P	*with*	4	5
NNS	*chopsticks*	5	6

Response

label	Text	Start	End
S	*she eats sushi with chopsticks*	1	6
NP	*she*	1	2
VP	*eats sushi with chopsticks*	2	6
V	*eats*	2	3
NP	*sushi with chopsticks*	3	6
N	*sushi*	3	4
PP	*with chopsticks*	4	6
P	*with*	4	5
NNS	*chopsticks*	5	6

$$R = 8/8 = 1$$
$$P = 8/9 = 0.89$$
$$F = 1.78/1.89 = 0.94$$

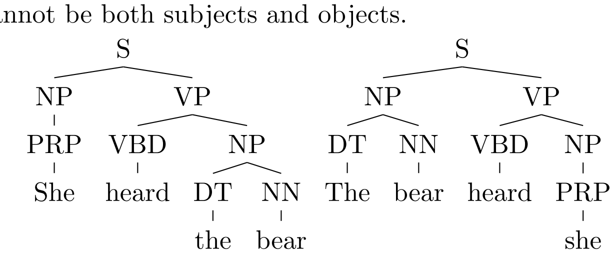
- Labeled and unlabeled scores are identical for this example.
- This is pretty high considering we made the only possible mistake (given the grammar above).
- Evaluation sometimes considers the number of sentences parsed entirely correctly.

9 Does PCFG parsing work?

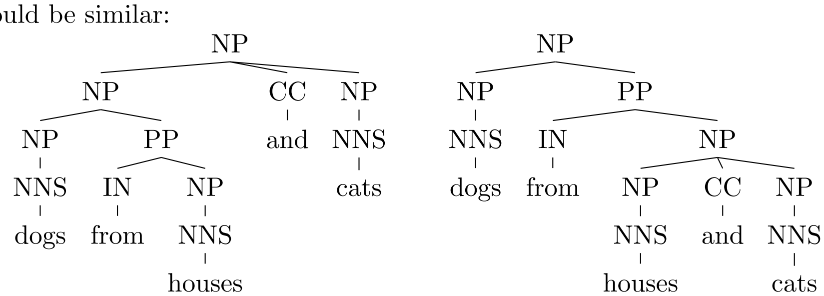
- A PCFG built from treebank probabilities scores $F = 0.72$
- Generally much better on short sentences, much worse on long ones.
- More remarkably, a PCFG estimated *from the test data* only achieves $F = 0.75$!
- Why isn't this better?
  - Given the PTB non-terminals, the context-free assumption is too strong.
  - If $P(NP \to NP\ PP) > P(VP \to VP\ PP)$, we will **always** choose NP attachment; otherwise, we will always choose VP attachment.
  - *She eats sushi with chopsticks*
    * $P(\text{NP-attach}) = P(S \to NP VP) \times P(VP \to V\, NP) \times P(NP \to NP\, PP) \times P(PP \to P\, NP)$
    * $P(\text{VP-attach}) = P(S \to NP VP) \times P(VP \to V PP P) \times P(VP \to V\, NP) \times P(PP \to P\, NP)$

9.1 Problems with PCFG parsing on the PTB

Substitutability Are NPs really substitutable? No, because many pronouns cannot be both subjects and objects.

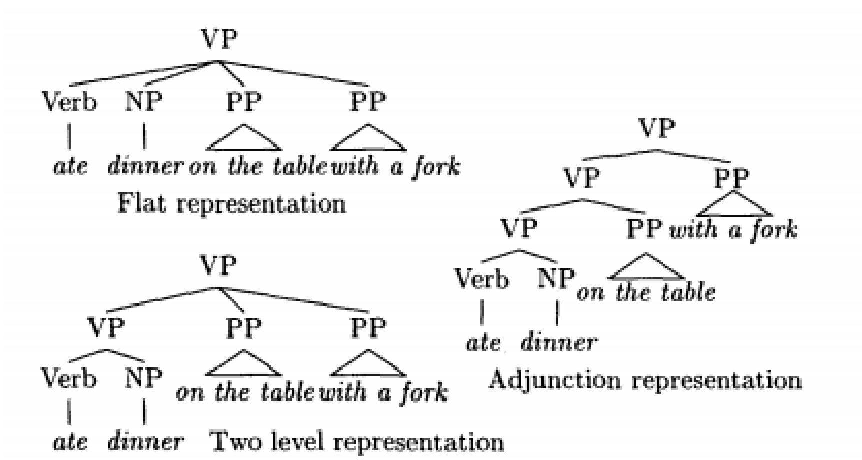


Semantic preferences In addition to grammatical constraints such as case marking, we have semantic preferences: for example, that conjoined entities should be similar:



- Which do you prefer?
- Could you build a PCFG to do the right thing?[3]
- (You can set the probabilities however you want!)

Subsumption There are several choices for annotating PP attachment



Mark Johnson (1998) shows that even though the two-level representation is chosen in the annotation, it can never be produced by a PCFG because the production is **subsumed**.

$$P(NP \to NP\ PP) = 0.112$$
$$P(NP \to NP\ PP\ PP) = 0.006$$
$$P(NP \to NP\ PP)P(NP \to NP\ PP) = (0.112)^2 = 0.013$$

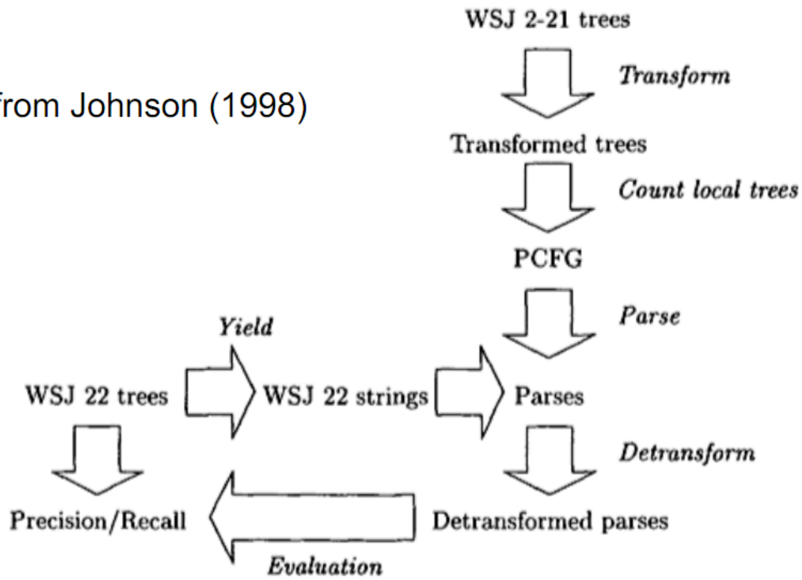
The probability of applying the NP $\to$ NP PP production twice is greater than the probability of the two-PP production, so this production will never appear in a PCFG parse. Johnson shows that 9% of all productions are subsumed and can be removed from the grammar!

10 Tree transformations

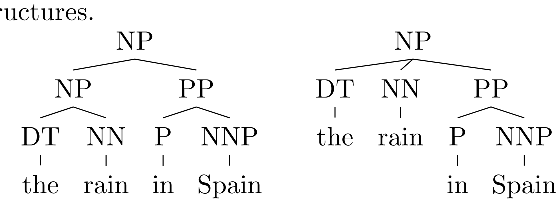
Johnson proposed a series of transformations to PTB trees that improve parsing accuracy.

[3]Example from Dan Klein

from Johnson (1998)

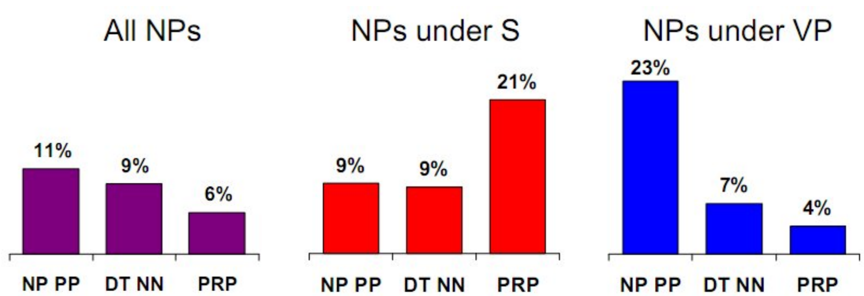


Flattening Johnson proposes "flattening" nested NPs to be more like VP structures.



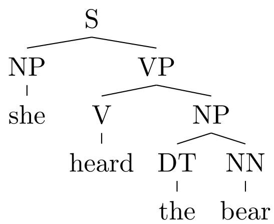
- will this increase or decrease the size of the grammar?

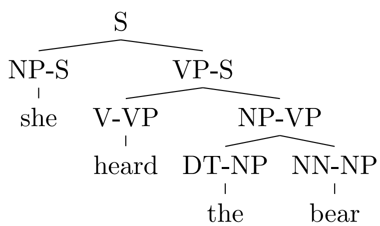
Parent annotation The expansion of an NP is highly dependent on its parent.



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

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

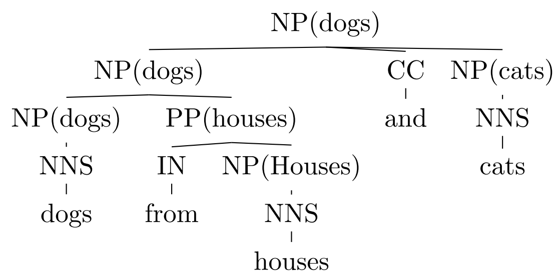
Overall, these transformations improve performance:

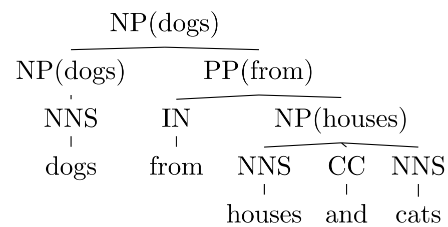
- Standard PCFG: 72% F-measure, 14,962 rules
- Parent-annotated PCFG: 80% F-measure, 22,773 rules
- In principle, parent annotation could have increased the grammar size much more drammatically, but many possible productions never occur, or are subsumed.

11 Lexicalization

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

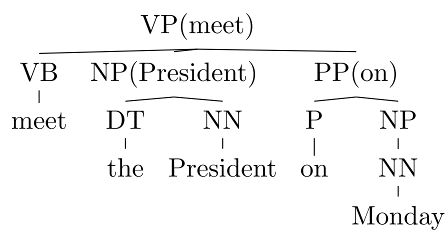
11.1 Example: coordination scope

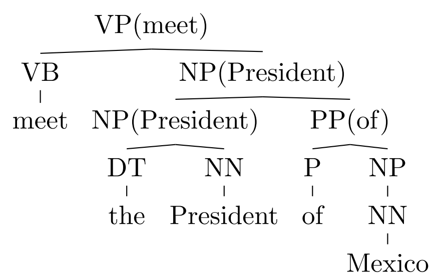




If $P(NP \rightarrow NP(\text{dogs})\ CC\ NP(\text{cats})) > (NP \rightarrow NP(\text{houses})\ CC\ NP(\text{cats}))$, we should get the right parse.

11.2 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.

11.3 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:

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

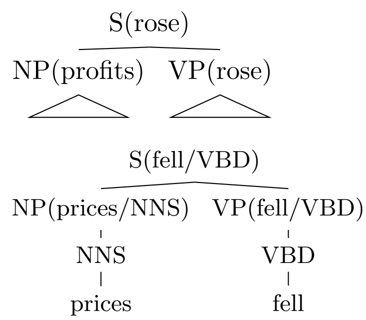
11.4 How to do it

- 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).

11.5 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 of constituent i
  - t_i is the syntactic category
  - $p(i)$ is the parent of node i
  - e.g.
    * $P(prices|NNS) = .013$
    * $P(prices|NNS, NP) = .013$
    * $P(prices|NNS, NP, S) = .025$
    * $P(prices|NNS, NP, S, VBD) = .052$
    * $P(prices|NNS, NP, S, VBD, fell) = .146$
  - Compute the rule probability: $P(r_i|t_i, s_i, t_{p(i)})$.
  - Score each production by the product of the rule probability and the head probabilities.
  - Apply standard CKY bottom-up parsing.



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%

11.5.1 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*

Clever smoothing is absolutely critical for lexicalized parsers.

11.5.2 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:

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

We have to tune $\lambda_1 \dots \lambda_4$, 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.

11.6 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,

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

- 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 $\bullet$ symbols.

VP(dumped)

VBD(dumped) NP(sacks) PP(into)

dumped sacks into the river

To model this rule, we would compute:

$P(VP(\mathit{dumped}, VBD) \rightarrow \bullet\ VBD(\mathit{dumped}, VBD)\ NP(\mathit{sacks}, NNS)\ PP(\mathit{into}, P)\ \bullet)$

- Here's the generative process:
  - Generate the head: $P(H \vert LHS) = P(VBD(\mathit{dumped}, VBD) \vert VP(\mathit{dumped}, VBD))$
  - Generate the left dependent: $P_L(\bullet \vert VP(\mathit{dumped}, VBD), VBD(\mathit{dumped}, VBD))$
  - Generate the right dependent: $P_R(NP(\mathit{sacks}, NNS) \vert VP(\mathit{dumped}, VBD), VBD(\mathit{dumped}, VBD))$

- 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.
- **Horizontal Markovization**: we condition only on the head
- Collins parser also conditions on a "distance" of each constituent from the head.

11.6.1 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).

11.6.2 The complexity of lexicalized parsing

- Straightforward lexicalized parsing is $\mathcal{O}(N^5 G)$, where
  - N is the length of the sentence

- G is the state space, equal to g^3 (cubic in the number of original non-terminals, because we condition on the head and the parent), times V^3 (cubic in the vocabulary size, for the same reason)

- Exhaustive search is totally infeasible; Collins and Charniak both use beam search to eliminate unpromising nodes from the chart.

- Eisner and Satta (2000, etc) give ways to parse more restricted classes of bilexical grammars in $O(N^4)$ or $O(N^3)$

11.7 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?