L95: Natural Language Syntax and Parsing
7) Parsing Accuracy

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge

Reminder...

We have looked at:

- grammars (PCFG, dependency, CCG)
- parsing algorithms (dynamic, deterministic, heuristic)
- parse scoring models (Bayesian, log-linear, cost-functions)
- methods for selecting n-best parses (beams, agendas)

But what do we need to do to make the parser as **accurate** as possible... ?

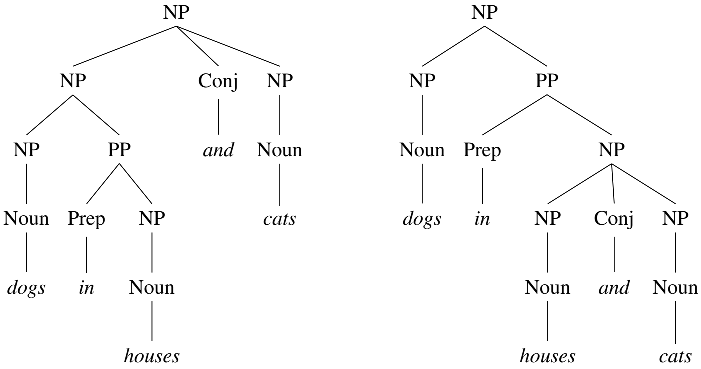
Reminder: PCFGs have some shortcomings

When we looked at PCFGs we noted two sources of inaccuracy:

- The **independence** assumption: unable to model structural dependency across the tree as a whole
  - The choice of how a non-terminal expands depend on the location in the parse tree.
  - In English, subject NPs are more likely to be pronouns ($\approx 90\%$), and objects NPs are more likely to be non-pronominal ($\approx 60\%$)
- Lack of **lexical specificity**: unable to model the structural behaviour specific to a lexical item
  - E.g. *VP*-attachment of *PP*s are more common in English
  - We will always get *some people like beer in cold glasses* wrong
  - Also lack of subcategorisation
  - And co-ordination

Reminder: PCFGs have some shortcomings

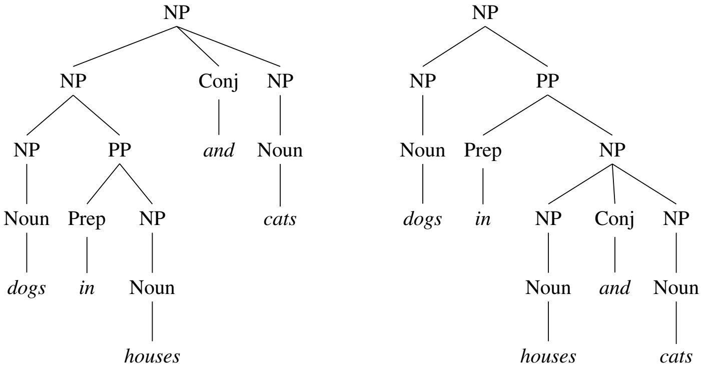
Lack of **lexical specificity**: these co-ordinated trees have the same probability...



From Jurafsky and Martin version 3, following Collins

Reminder: PCFGs have some shortcomings

Lack of **lexical specificity**: these co-ordinated trees have the same probability...



From Jurafsky and Martin version 3, following Collins

Today will we look as how to get around these issues.

Relax independence by splitting non-terminals

For the pronoun issue, intuition is that we need more NP rules:

Relax independence by splitting non-terminals

For the pronoun issue, intuition is that we need more NP rules:
instead of $NP \rightarrow PRP$ we need two rules:

- $NPsubject \rightarrow PRP$
- $NPobject \rightarrow PRP$

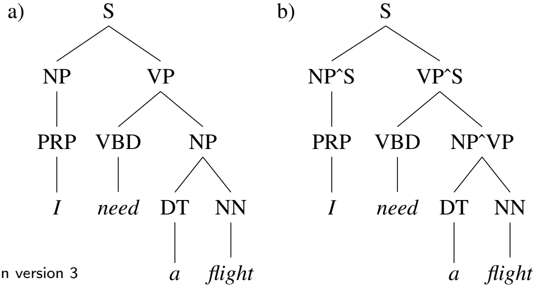
How can we implement this without a semantic treebank?

Relax independence by splitting non-terminals

For the pronoun issue, intuition is that we need more NP rules:
instead of *NP* → *PRP* we need two rules:

- *NPsubject* → *PRP*
- *NPobject* → *PRP*

How can we implement this without a semantic treebank? by annotating non-terminals with their parent nodes



From Jurafsky and Martin version 3

Parent annotation helps in several scenarios

- Other examples of parent annotation:

  e.g. differentiating between **adverbs** by annotating pre-terminals with their parents

  e.g. **subordinating conjunctions**, *while*, *as*, *if*, occur under *S*

- Where parent annotation can't help we could split on other features (i.e. hand write rules for specific feature scenarios)

- See `https://nlp.stanford.edu/manning/papers/unlexicalized-parsing.pdf` for some discussion

A **trade-off** between splitting and training

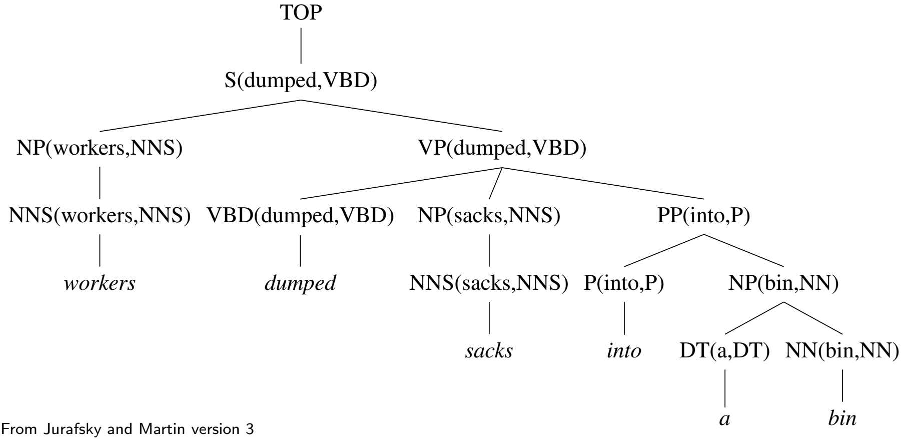
- Splitting non-terminals increases the grammar size
- Increased grammar size means less data per rule instance for MLE
- **split and merge** techniques automatically search for the optimal splits by **maximising the likelihood** of the training set (e.g. **Petrov et al. 2006**)

non-terminal splitting example in class

Lexicalised-PCFGs include lexical info in the grammar

Collins and **Charniak** parsers use lexicalised-PCFGs

- Lexicalisation can include both the **head** word token and its part-of-speech



From Jurafsky and Martin version 3

Lexicalised-PCFGs include **lexical** info in the grammar

- For each rule one of the RHS daughters is the **head**
- The head information for the LHS of the rule is the same as the RHS head
- **Pre-terminal rules** always have a **probability of** 1
- All other rule probabilities need to be calculated ...

  ... **but the data available per rule is now very sparse**

Lexicalised-PCFGs include **lexical** info in the grammar

- For each rule one of the RHS daughters is the **head**
- The head information for the LHS of the rule is the same as the RHS head
- **Pre-terminal rules** always have a **probability of** 1
- All other rule probabilities need to be calculated ...

  ... **but the data available per rule is now very sparse**

Collins handles sparsity by **generating** the RHS of rules

- RHS of every rule consists of a **head** plus all the non-terminals to the head's **left** and all the non-terminals to the head's **right**
  $LHS \rightarrow L_m \dots L_1\ H\ R_1 \dots R_n$

- To use a rule we:

- first **generate the head**,
- then all the **left dependents** from the head outwards
- and finally all the **right dependents** from the head outwards

- We imagine a $STOP$ non-terminal at the edges of the rule
  $LHS \rightarrow STOP\ L_m \dots L_1\ H\ R_1 \dots R_n\ STOP$

Rule probability is the **product** of all generated pieces

- Remember that for PCFGs: $P(A \rightarrow B) = P(B|A)$

- For lexicalised PCFGs: $A \rightarrow STOP\ L_m \ldots L_1\ H\ R_1 \ldots R_n\ STOP$

- The probability of the head H with associated word h_w and tag h_t given the parent, A is:

  $P(H(w_h, t_h)) = P(H(h_w, h_t)|A, h_w, h_t)$

- The probability of modifiers to the left of the head is:

  $$\prod_{i=1}^{m+1} P(L_i(lw_i, lt_i)|A, H, h_w, h_t)$$

- The probability of modifiers to the right of the head is:

  $$\prod_{i=1}^{n+1} P(R_i(rw_i, rt_i)|A, H, h_w, h_t)$$

  where $L_{m+1} = STOP$ and $R_{n+1} = STOP$

lexicalised-PCFG rule probability estimation in class

Collins models have **other conditional features**

- Collins 1 includes a **distance metic** in the conditional probabilities
- Collins 2 includes conditioning on **subcategorisation** and **argument/adjunct**

- In training Collin's interpolates three models:
- fully lexicalised (conditioning on the head word and tag),
- just the head tag
- unlexicalized

Remember **Coarse-to-fine** strategy, Charniak

We can now understand better Charniak's **coarse-to-fine** parsing strategy:

1. produce a parse forest using simple version of the grammar
   i.e. find possible parses using coarse-grained non-terminals, e.g. *VP*
2. refine most promising of coarse-grained parses using complex grammar
   i.e with feature-based, lexicalised non-terminals, e.g. $VP[buys/VBZ]$

- **Coarse-grained step** can be **efficiently parsed** using e.g. CKY
- But the simple grammar **ignores contextual features** so best parse might not be accurate
- **Output a pruned packed parse** forest for the parses generated by the simple grammar (using a beam threshold)
- **Evaluate remaining parses with complex grammar** (i.e. each coarse-grained state is split into several fine-grained states)