Statistical Parsing

Statistical parsing

- Over the last 12 years statistical parsing has succeeded wonderfully!
- NLP researchers have produced a range of (often free, open source) statistical parsers, which can parse *any sentence* and *often get most of it correct*
- These parsers are now a commodity component
- The parsers are still improving year-on-year.

Statistical Parsing

- Basic idea
  - Start with a treebank
    - a collection of sentences with syntactic annotation, i.e., already-parsed sentences
  - Examine which parse trees occur frequently
  - Extract grammar rules corresponding to those parse trees, estimating the probability of the grammar rule based on its frequency
- That is, we'll have a CFG augmented with probabilities

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree (presumably the right one).
- These are generally created
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.

Penn Treebank

- Penn TreeBank is a widely used treebank.

  - Most well known is the Wall Street Journal section of the Penn TreeBank.
    - 1 M words from the 1987-1989 Wall Street Journal.

```
( (S ('' '')
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Treebank Grammars

1

Treebanks implicitly define a grammar for the language covered in the treebank.

2

Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar.

3

Not complete, but if you have decent size corpus, you'll have a grammar with decent coverage.

Treebank Grammars

- Such grammars tend to be very flat due to the fact that they tend to avoid recursion.
  - To ease the annotators burden
- For example, the Penn Treebank has 4500 different rules for VPs. Among them...

$$
\begin{aligned}
\mathrm{VP} &\rightarrow \mathrm{VBD}\ \mathrm{PP} \\
\mathrm{VP} &\rightarrow \mathrm{VBD}\ \mathrm{PP}\ \mathrm{PP} \\
\mathrm{VP} &\rightarrow \mathrm{VBD}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP} \\
\mathrm{VP} &\rightarrow \mathrm{VBD}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP}\ \mathrm{PP}
\end{aligned}
$$

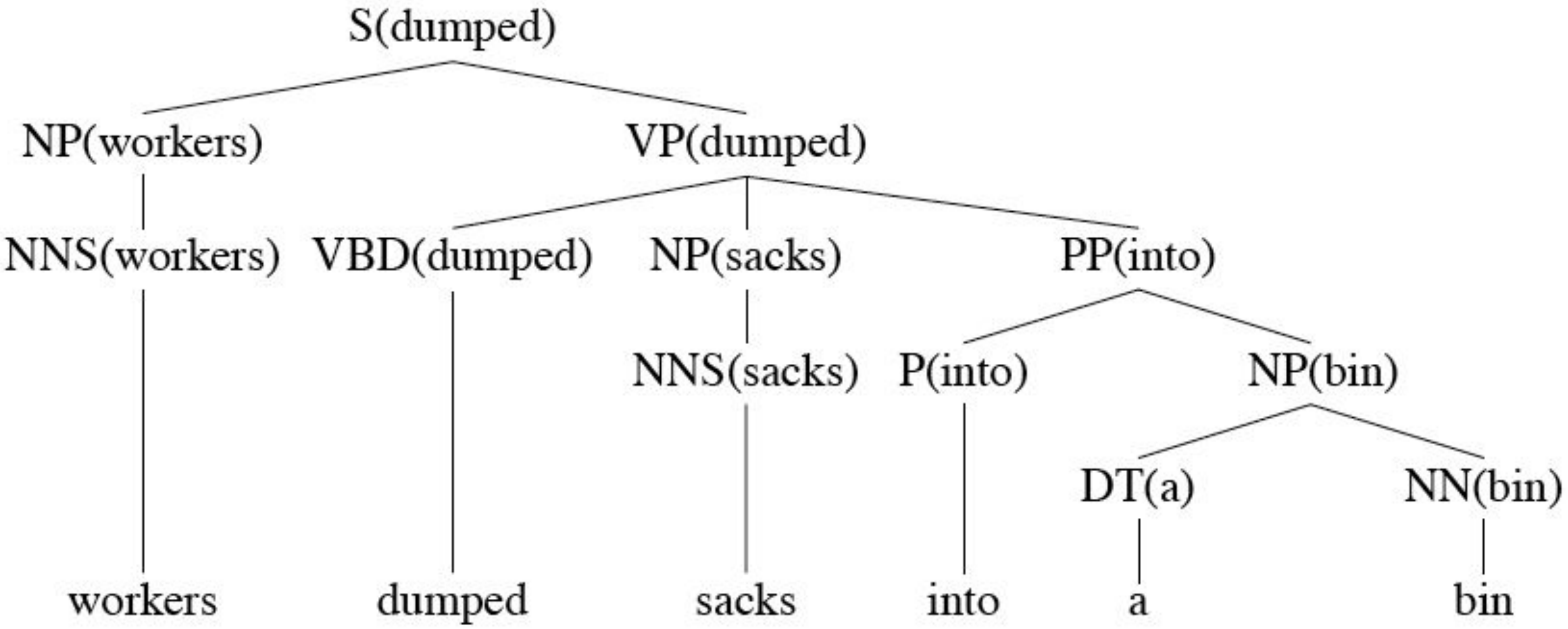
Heads in Trees

Finding heads in treebank trees is a task that arises frequently in many applications.

Particularly important in statistical parsing

We can visualize this task by annotating the nodes of a parse tree with the heads of each corresponding node.

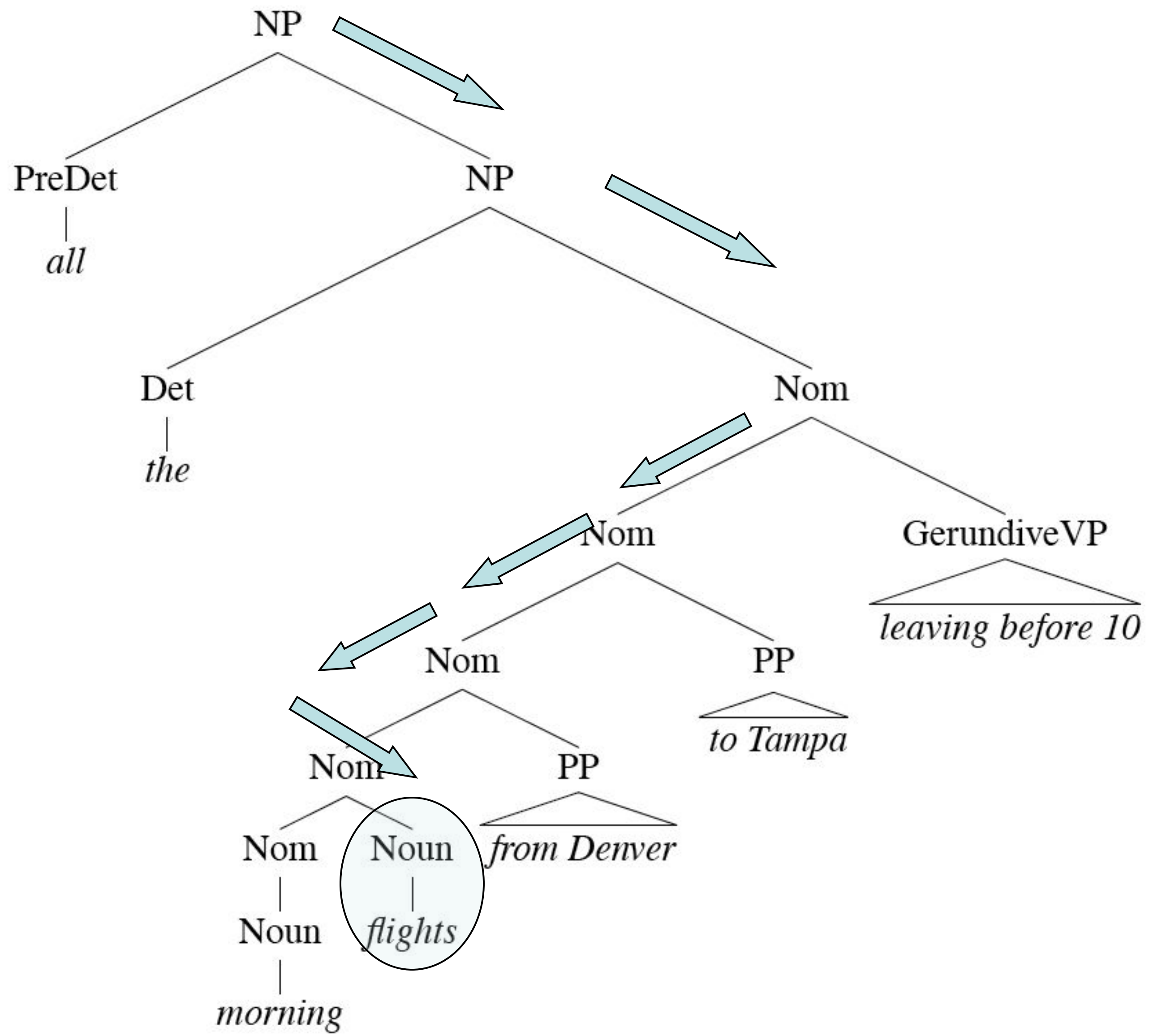
Lexically Decorated Tree



Head Finding

- The standard way to do head finding is to use a simple set of tree traversal rules specific to each non-terminal in the grammar.

Noun Phrases



Treebank Uses

- Treebanks (and headfinding) are particularly critical to the development of statistical parsers
- Also valuable to *Corpus Linguistics*
  - Investigating the empirical details of various constructions in a given language
    - How often do people use various constructions and in what contexts...
    - Do people ever say ...

Probabilistic CFGs

1 The probabilistic model

- Assigning probabilities to parse trees

2 Training the model (Learning)

- Acquiring estimates for the probabilities specified by the model

3 Parsing with probabilities (Decoding)

- Given an input sentence, using the model to find the best (or n-best) tree for the input

Rule Probabilities

- So... What's the probability of a rule?
- Start at the top...
  - A tree should have an *S* at the top. So given that we know we need an *S*, we can ask about the probability of each particular *S* rule in the grammar.
    - That is P(particular rule | S)
- So in general we need

$$P(\alpha \to \beta \mid \alpha)$$

For each rule in the grammar

Probability Model (1.1)

- The probability of a word sequence (sentence) is the probability of its tree in the unambiguous case.
- In the ambiguous case, it's the sum of the probabilities of the trees.
- Since we can use the probability of the tree(s) as a proxy for the probability of a sentence…
  - PCFGs give us an alternative to *N*-gram models as a kind of language model.

Learning

- What if you have a corpus but don’t have a treebank to get the counts from?
- Parse the corpus with a non-probabilistic grammar and collect counts for the rules that get used.
- Normalize

- Parse the corpus with a non-probabilistic grammar and collect counts for the rules that get used.
- Prorate the counts by the probability of the parse that it comes from.
- Normalize; update; iterate.

Parsing (Decoding)

- So to get the best (most probable) parse for a given input

1. Enumerate all the trees for a sentence
2. Assign a probability to each using the model
3. Return the argmax

Formal Definition of PCFG

- A PCFG consists of
  - A set of terminals $\{w_k\}$, k = 1,….,V

    $\{w_k\}$ = { child, teddy, bear, played…}
  - A set of non-terminals $\{N^i\}$, i = 1,…,n

    $\{N_i\}$ = { NP, VP, DT…}
  - A designated start symbol N^1
  - A set of rules $\{N^i \rightarrow \zeta^j\}$, where ζ^j is a sequence of terminals & non-terminals

    NP $\rightarrow$ DT NN
  - A corresponding set of rule probabilities

Rule Probabilities

- Rule probabilities are such that

$$\forall i \quad \sum_{\mathrm{i}} \mathrm{P}(\mathrm{N}^{i} \rightarrow \zeta^{j}) = 1$$

E.g., P(NP → DT NN) = 0.2
P(NP → NN) = 0.5
P(NP → NP PP) = 0.3

- P(NP → DT NN) = 0.2
  - Means 20 % of the training data parses use the rule NP → DT NN

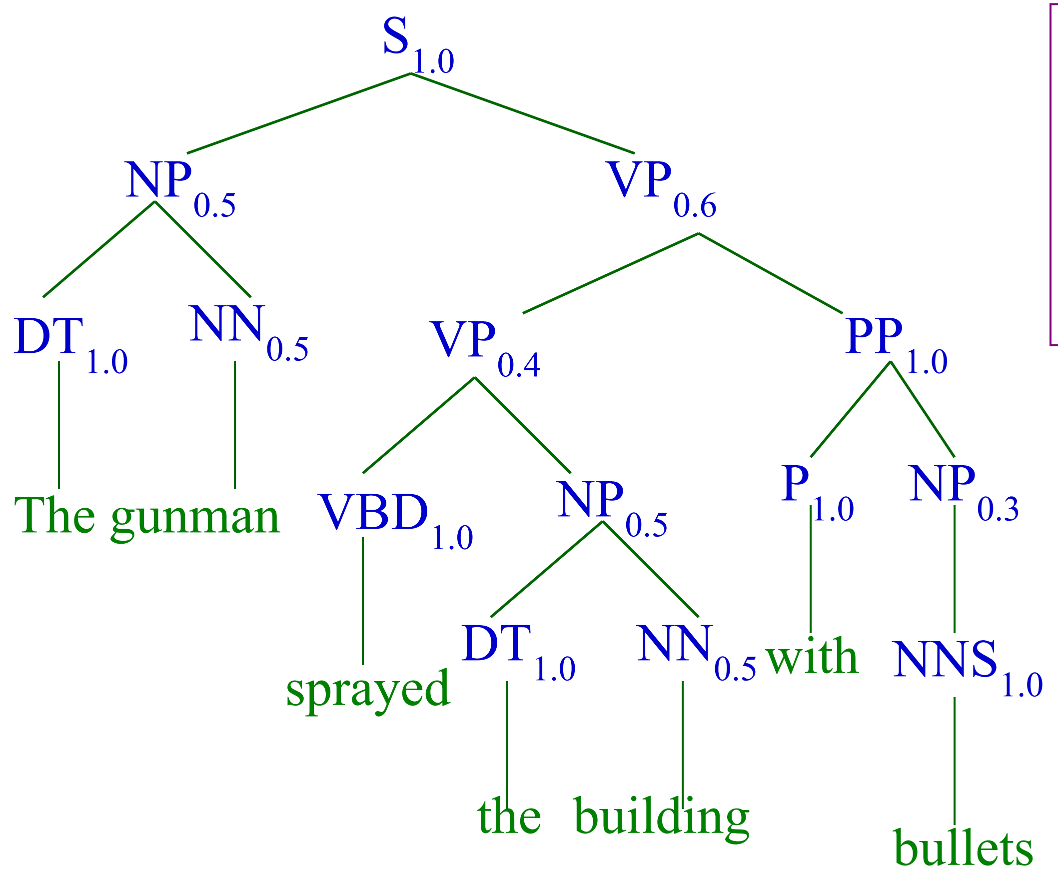
Example PCFG Rules & Probabilities

- S → NP VP 1.0
- NP → DT NN 0.5
- NP → NNS 0.3
- NP → NP PP 0.2
- PP → P NP 1.0
- VP → VP PP 0.6
- VP → VBD NP 0.4

- DT → the 1.0
- NN → gunman 0.5
- NN → building 0.5
- VBD → sprayed 1.0
- NNS → bullets 1.0

Example Parse t_1`

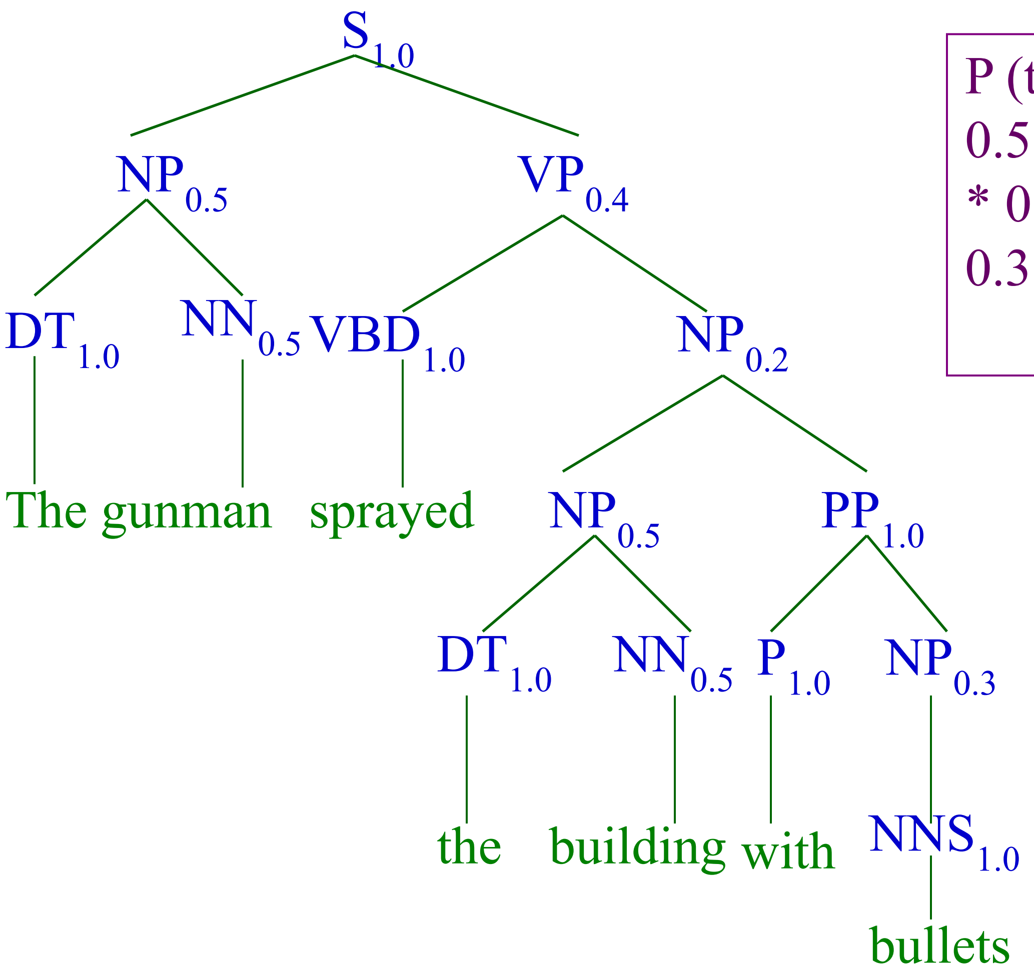
- The gunman sprayed the building with bullets.



P (t_1) = 1.0 * 0.5 * 1.0 * 0.5 * 0.6 * 0.4 * 1.0 * 0.5 * 1.0 * 0.5 * 1.0 * 1.0 * 0.3 * 1.0 = 0.00225

Another Parse t_2

- The gunman sprayed the building with bullets.



$P(t_2) = 1.0 * 0.5 * 1.0 * 0.5 * 0.4 * 1.0 * 0.2 * 0.5 * 1.0 * 0.5 * 1.0 * 1.0 * 0.3 * 1.0 = 0.0015$