
Advanced Natural Language Processing

Lecture 14

Statistical Parsing (I)

Frank Keller (slides by Philipp Koehn)

25 October 2011



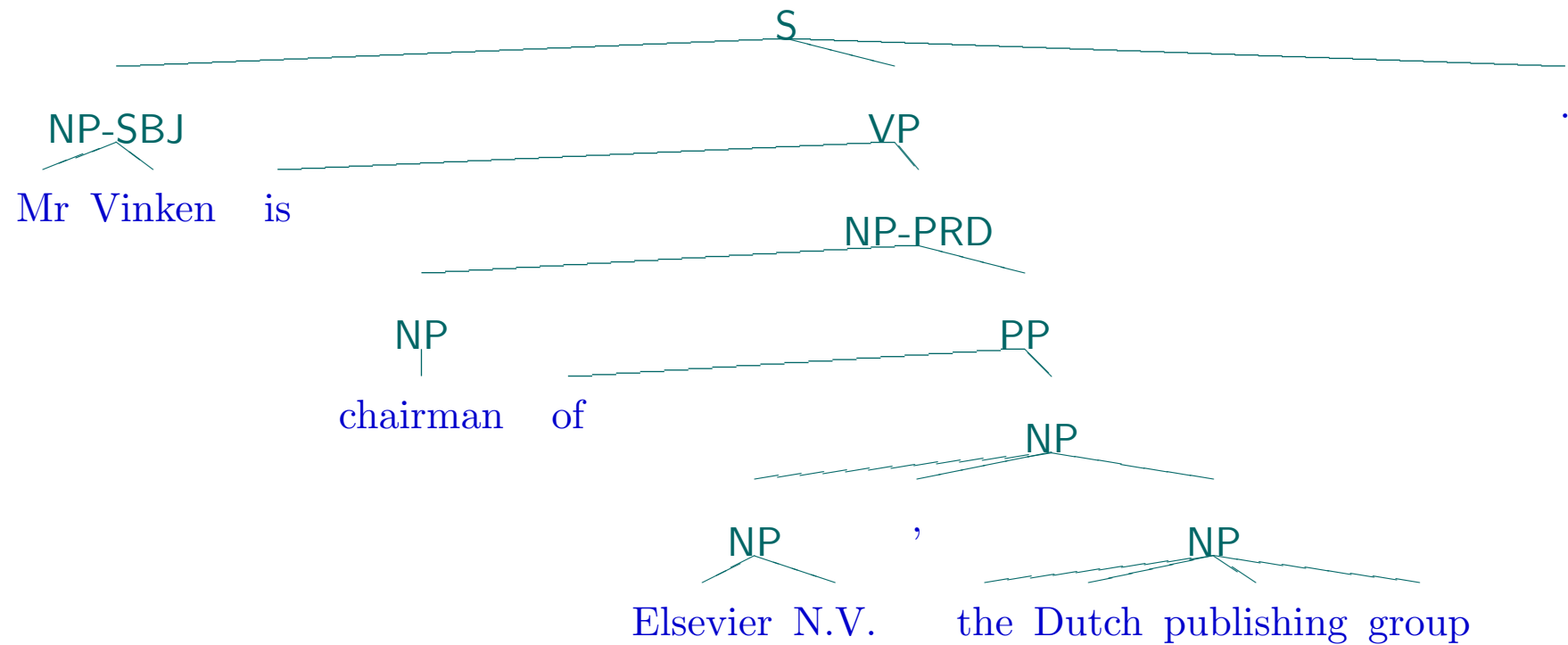
Parsing

- Task: build the syntactic tree for a sentence
- Grammar formalism
 - phrase structure grammar
 - context-free grammar
- Parsing algorithm: CYK (chart) parsing
- Open problems
 - where do we get the grammar from?
 - how do we resolve ambiguities

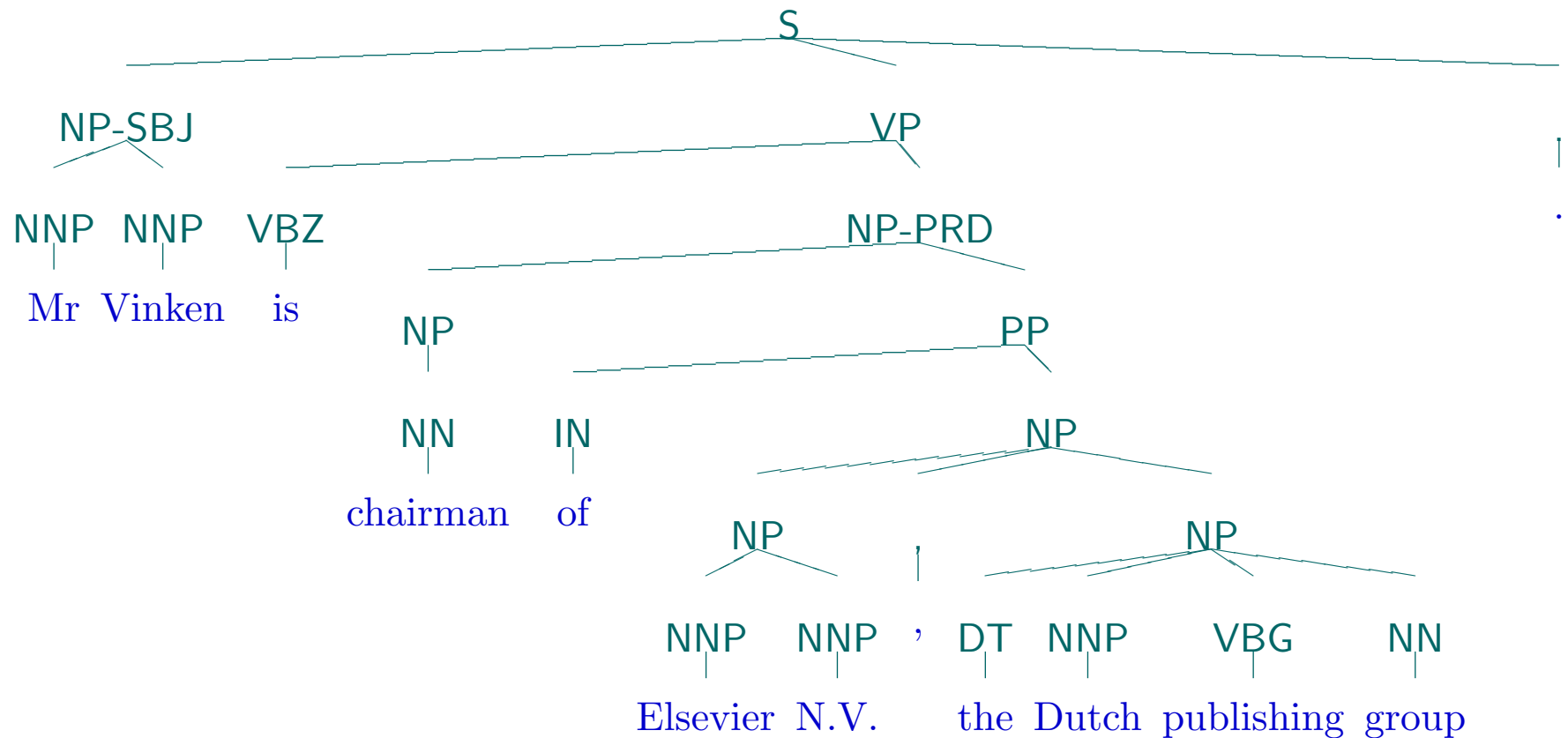
Penn Treebank

- Penn treebank: English sentences annotated with syntax trees
 - built at the University of Pennsylvania
 - 40,000 sentences, about a million words
 - real text from the Wall Street Journal
- Similar treebanks exist for other languages
 - German
 - French
 - Spanish
 - Arabic
 - Chinese
 - etc.

Sample Syntax Tree



Sample Tree with Part-of-Speech



Learning a Grammar from the Treebank

- Context-free grammar: we have rules in the form

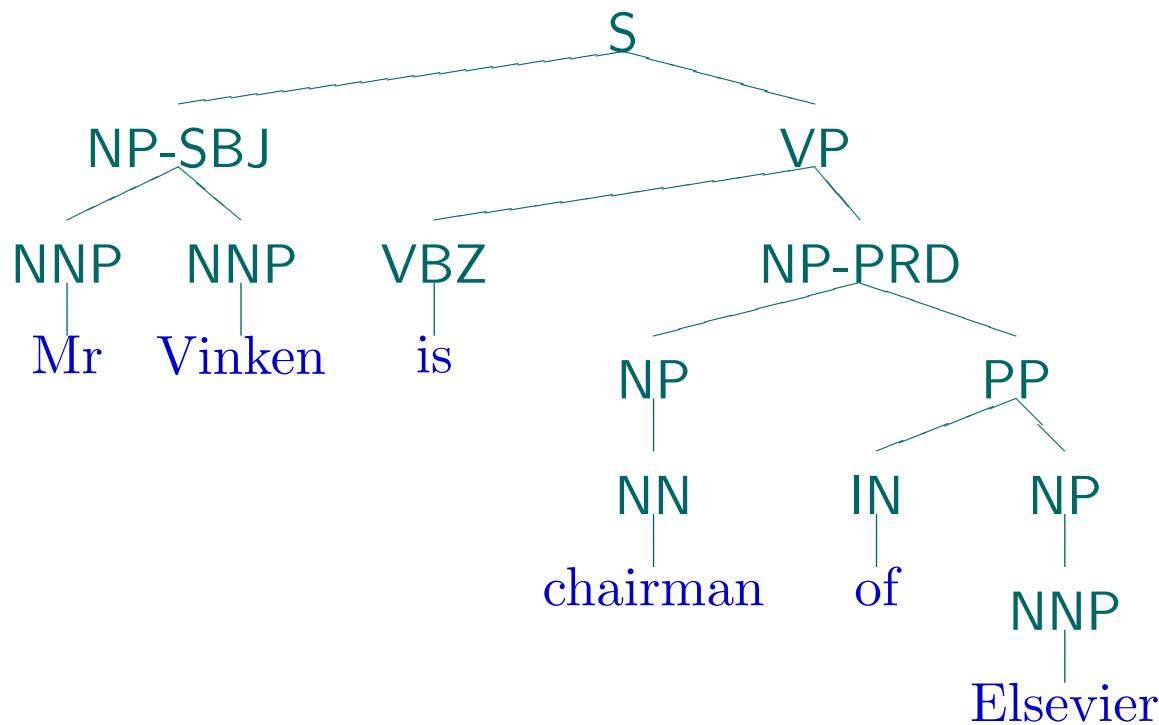
$$S \rightarrow \text{NP-SBJ VP}$$

- We can collect these rules from the treebank
- We can even estimate probabilities for rules

$$p(S \rightarrow \text{NP-SBJ VP} | S) = \frac{\text{count}(S \rightarrow \text{NP-SBJ VP})}{\text{count}(S)}$$

⇒ Probabilistic context-free grammar (PCFG)

Rules Applications to Build Tree



- S → NP-SBJ VP
- NP-SBJ → NNP NNP
- NNP → Mr
- NNP → Vinken
- VP → VBZ NP-PRD
- VBZ → is
- NP-PRD → NP PP
- NP → NN
- NN → chairman
- PP → IN NP
- IN → of
- NP → NNP
- NNP → Elsevier

Compute Probability of Tree

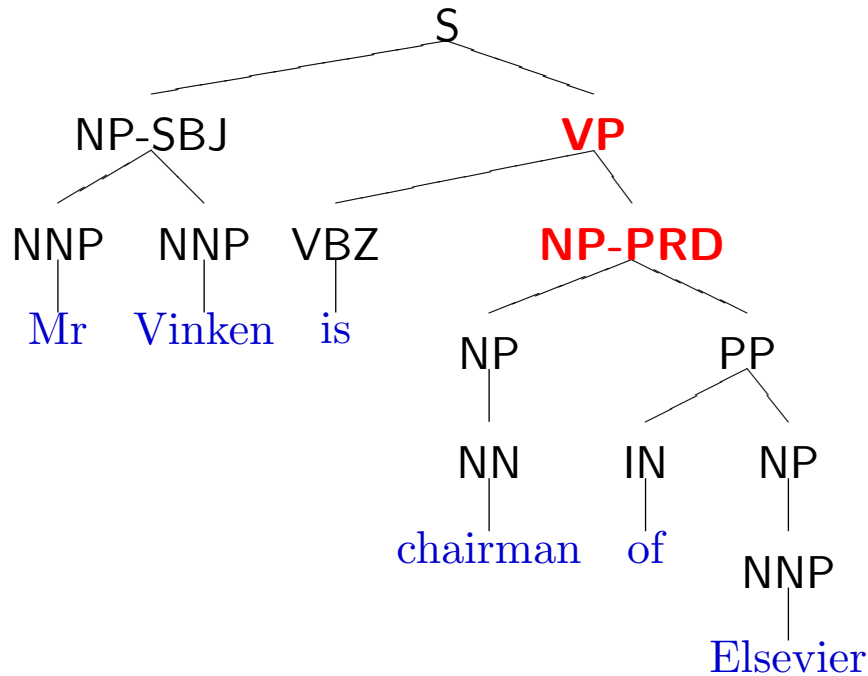
- Probability of a tree is the product of the probabilities of the rule applications:

$$p(\text{tree}) = \prod_i p(\text{rule}_i)$$

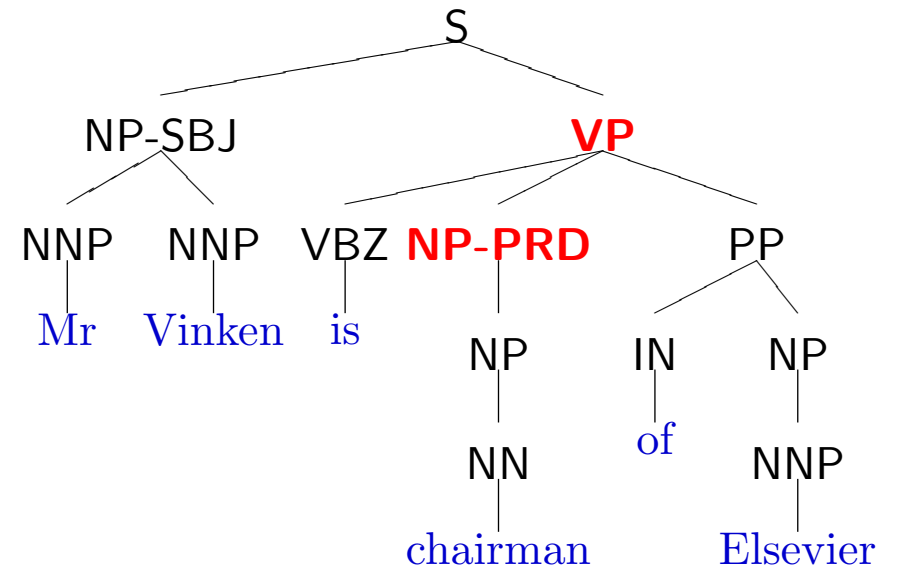
- We assume that all rule applications are independent of each other

$$\begin{aligned} p(\text{tree}) &= p(S \rightarrow \text{NP-SBJ VP} | S) \times \\ & p(\text{NP-SBJ} \rightarrow \text{NNP NNP} | \text{NP-SBJ}) \times \\ & \dots \times \\ & p(\text{NNP} \rightarrow \text{Elsevier} | \text{NNP}) \end{aligned}$$

Prepositional Phrase Attachment Ambiguity



PP attached to NP-PRD



PP attached to VP

Rule Applications

S → NP-SBJ VP
NP-SBJ → NNP NNP
NNP → Mr
NNP → Vinken
VP → VBZ NP-PRD
VBZ → is
NP-PRD → NP PP
NP → NN
NN → chairman
PP → IN NP
IN → of
NP → NNP
NNP → Elsevier

PP attached to NP-PRD

S → NP-SBJ VP
NP-SBJ → NNP NNP
NNP → Mr
NNP → Vinken
VP → VBZ NP-PRD PP
VBZ → is
NP-PRD → NP
NP → NN
NN → chairman
PP → IN NP
IN → of
NP → NNP
NNP → Elsevier

PP attached to VP

Difference in Probability

- PP attachment to NP-PRD is preferred if

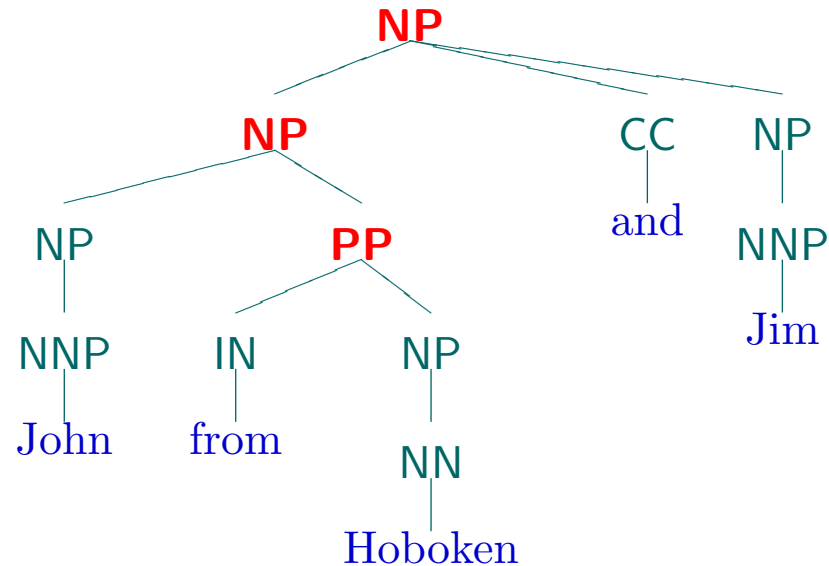
$$p(\text{VP} \rightarrow \text{VBZ NP-PRD} | \text{VP}) \times p(\text{NP-PRD} \rightarrow \text{NP PP} | \text{NP-PRD})$$

is larger than

$$p(\text{VP} \rightarrow \text{VBZ NP-PRD PP} | \text{VP}) \times p(\text{NP-PRD} \rightarrow \text{NP} | \text{NP-PRD})$$

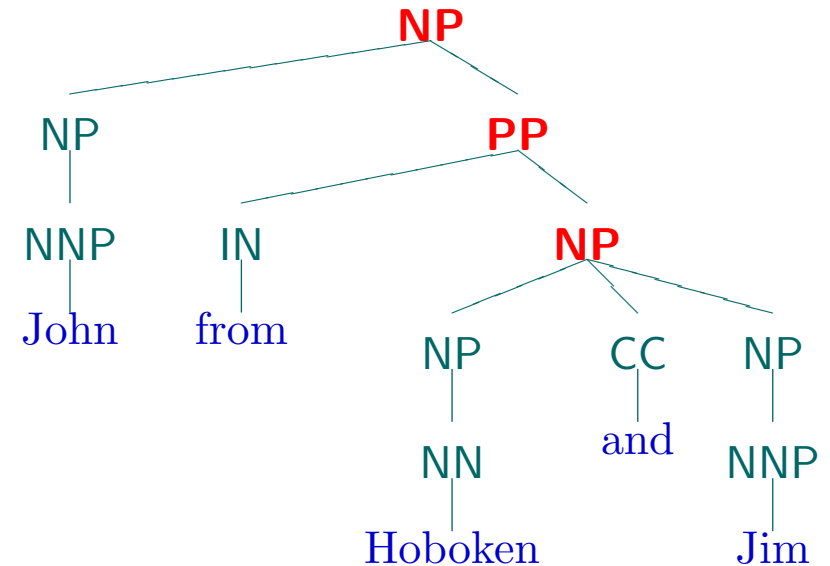
- Is this too general?

Scope Ambiguity



correct:

and connects John and Jim



false:

and connects Hoboken and Jim

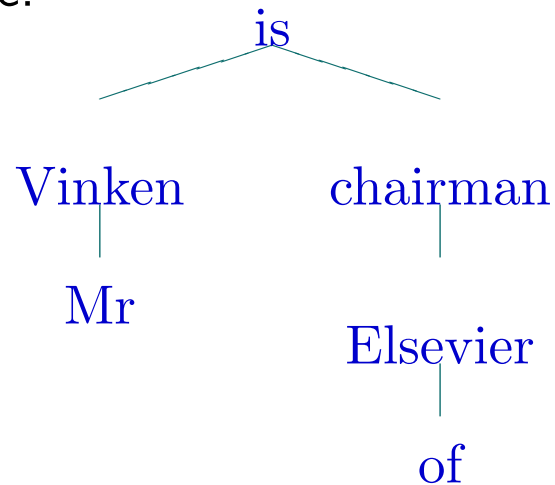
However: the same rules are applied

Weakness of PCFG

- Independence assumption too strong
- Non-terminal rule applications do not use lexical information
- Not sufficiently sensitive to structural differences beyond parent/child node relationships

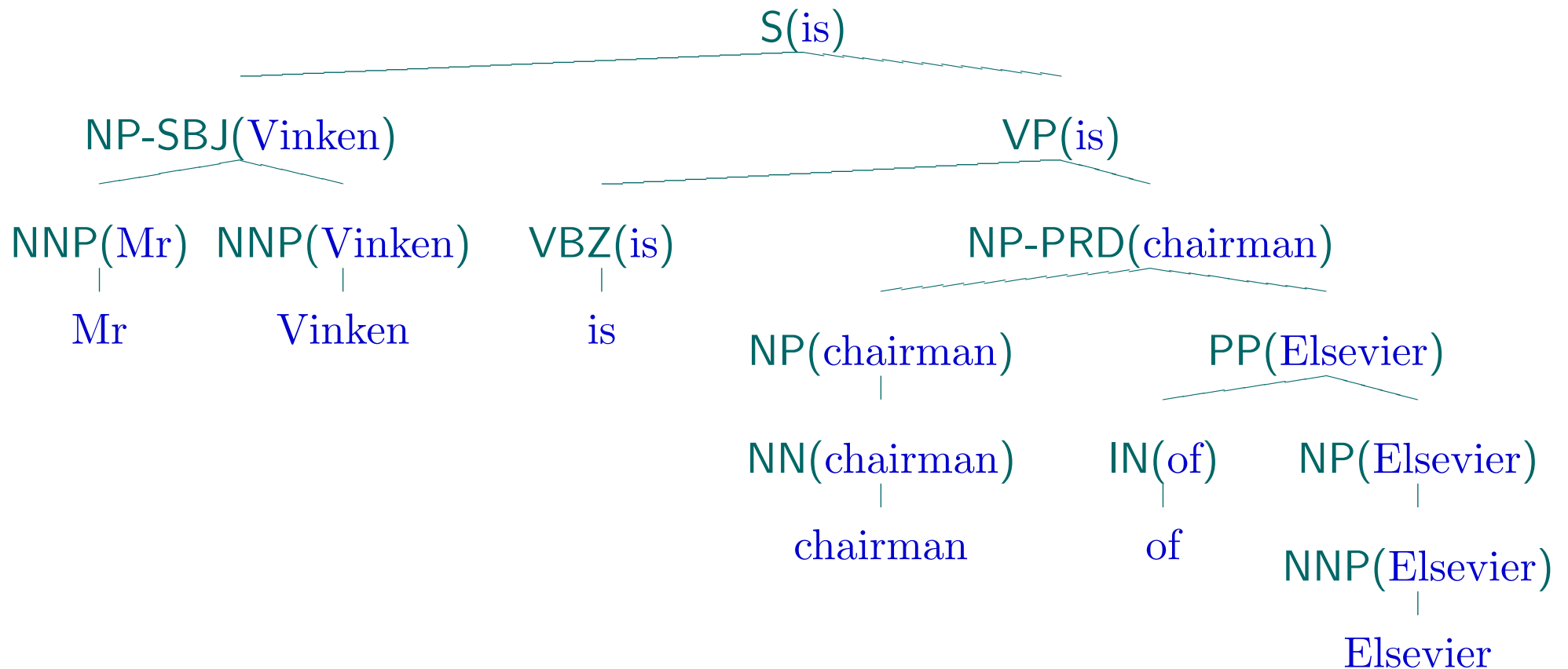
Head Words

- Recall dependency structure:



- Direct relationships between words, some are the head of others (see also Tree Adjoining Grammar)

Adding Head Words to Trees



Head words in rules

- Each context-free rule has one head child that is the head of the rule
 - $S \rightarrow NP VP$
 - $VP \rightarrow VBZ NP$
 - $NP \rightarrow DT NN NN$
- Parent receives head word from head child
- Head children are not marked in the Penn treebank, but they are easy to recover using simple rules

Recovering heads

- Rule for recovering heads for NPs
 - if rule contains NN, NNS or NNP, choose rightmost NN, NNS or NNP
 - else if rule contains a NP, choose leftmost NP
 - else if rule contains a JJ, choose rightmost JJ
 - else if rule contains a CD, choose rightmost CD
 - else choose rightmost child
- Examples
 - NP → DT NNP NN
 - NP → NP CC NP
 - NP → NP PP
 - NP → DT JJ
 - NP → DT

Using head nodes

- PP attachment to NP-PRD is preferred if

$$p(\text{VP}(\text{is}) \rightarrow \text{VBZ}(\text{is}) \text{NP-PRD}(\text{chairman}) | \text{VP}(\text{is}))$$

$$\times p(\text{NP-PRD}(\text{chairman}) \rightarrow \text{NP}(\text{chairman}) \text{PP}(\text{Elsevier}) | \text{NP-PRD}(\text{chairman}))$$

is larger than

$$p(\text{VP}(\text{is}) \rightarrow \text{VBZ}(\text{is}) \text{NP-PRD}(\text{chairman}) \text{PP}(\text{Elsevier}) | \text{VP}(\text{is}))$$

$$\times p(\text{NP-PRD}(\text{chairman}) \rightarrow \text{NP}(\text{chairman}) | \text{NP-PRD}(\text{chairman}))$$

- Scope ambiguity: combining **Hoboken** and **Jim** should have low probability

$$p(\text{NP}(\text{Hoboken}) \rightarrow \text{NP}(\text{Hoboken}) \text{CC}(\text{and}) \text{NP}(\text{John}) | \text{VP}(\text{Hoboken}))$$

Sparse data concerns

- How often will we encounter

NP(Hoboken) → NP(Hoboken) CC(and) NP(John)

- ... or even

NP(Jim) → NP(Jim) CC(and) NP(John)

- If not seen in training, probability will be zero

Sparse data: Dependency relations

- Instead of using a complex rule

$\text{NP}(\text{Jim}) \rightarrow \text{NP}(\text{Jim}) \text{CC}(\text{and}) \text{NP}(\text{John})$

- ... we collect statistics over dependency relations

head word	head tag	child node	child tag	direction
Jim	NP	and	CC	left
Jim	NP	John	NP	left

- first generate child tag: $p(\text{CC}|\text{NP}, \text{Jim}, \text{left})$
- then generate child word: $p(\text{and}|\text{NP}, \text{Jim}, \text{left}, \text{CC})$

Sparse Data: Interpolation

- Use of interpolation with back-off statistics (recall: language modeling)
- Generate child tag

$$p(\text{CC}|\text{NP}, \text{Jim}, \text{left}) = \lambda_1 \frac{\text{count}(\text{CC}, \text{NP}, \text{Jim}, \text{left})}{\text{count}(\text{NP}, \text{Jim}, \text{left})} + \lambda_2 \frac{\text{count}(\text{CC}, \text{NP}, \text{left})}{\text{count}(\text{NP}, \text{left})}$$

- With $0 \leq \lambda_1 \leq 1$, $0 \leq \lambda_2 \leq 1$, $\lambda_1 + \lambda_2 = 1$

Sparse Data: Interpolation (2)

- Generate child word

$$\begin{aligned} p(\text{and}|\text{CC}, \text{NP}, \text{Jim}, \text{left}) &= \lambda_1 \frac{\text{count}(\text{and}, \text{CC}, \text{NP}, \text{Jim}, \text{left})}{\text{count}(\text{CC}, \text{NP}, \text{Jim}, \text{left})} \\ &+ \lambda_2 \frac{\text{count}(\text{and}, \text{CC}, \text{NP}, \text{left})}{\text{count}(\text{CC}, \text{NP}, \text{left})} \\ &+ \lambda_3 \frac{\text{count}(\text{and}, \text{CC}, \text{left})}{\text{count}(\text{CC}, \text{left})} \end{aligned}$$

- With $0 \leq \lambda_1 \leq 1$, $0 \leq \lambda_2 \leq 1$, $0 \leq \lambda_3 \leq 1$, $\lambda_1 + \lambda_2 + \lambda_3 = 1$

What also helps

- Adding a count for distance from head word
- Part-of-speech of the head word and the child word also useful
- Improving tags
 - instead of general **VB**, distinguish between intransitive verb phrases **Vi**, and transitive verb phrases **Vt**
 - distinguish between complements (required attachments, e.g. object of a transitive verb) and adjuncts (optional attachments, e.g. *yesterday*)
- Not only use parent tag, but also grand-parent tag
- Create n-best list of best parse trees, re-score

Parsing algorithm

- Efficient parsing algorithm is tricky
- Algorithm is similar to chart parsing, as presented
- Impossible to search entire space of possible parse trees

→ rest cost estimation, pruning

Performance

- Performance typically measured in recall/precision of dependency relations
 - PCFG: 74.8%/70.6%
 - using lexical dependencies: 85.7%/85.3%
 - state of the art (e.g., Charniak and Johnson): 91.8%/91.0%
- Core sentence structure (complements, NP chunks) recovered with over 90% accuracy
- Attachment ambiguities involving adjuncts are resolved with much lower accuracy (~80% for PP attachment, ~50-60% for coordination)

Note: numbers quoted from lecture 4 [Parsing and Syntax II](#) of MIT class [6.891 Natural Language Processing](#) by [Michael Collins](#) (2005)