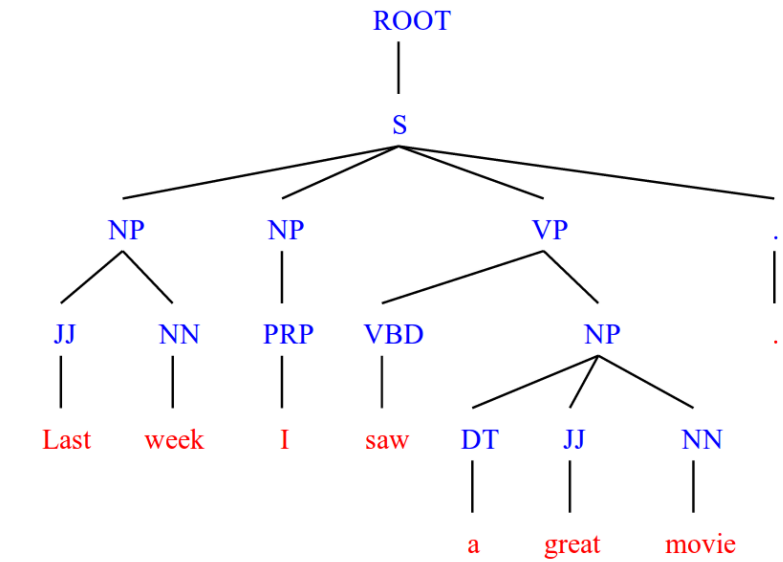
ENLP
Constituent Parsing

Malte Ahl

Seminar Presentation
Organizers:
Prof. Dr. Hannah Bast, assisted by Elmar Haußmann.
Chair for Algorithms and Data Structures
Universität Freiburg

December 21, 2011

Example constituent tree.




```
(ROOT
  (S
    (NP (JJ Last) (NN week))
    (NP (PRP I))
    (VP (VBD saw)
      (NP (DT a) (JJ great) (NN movie)))
    (. .)))
```

An example of a constituent tree (Stanford parser).

Motivation

Benefit of parsing trees:

1. Description of parsing problems.
2. Description of solution to solve e.g. ambiguity.
3. The second version of Penn Treebank and many NLP applications are using parsing trees as their output.

Definition constituents

1. **Constituents** are in most papers considered as Part-of-speech tags arranged in a tree structure.
2. A **constituent tree** of a sentence is a n-ary tree with a root element, (e.g labeled with 'TOP' or 'ROOT'). A terminal node of a constituent tree represents a single word or a sign of the regarding sentence. Such a tree is also called nonlexicalized parse tree.
3. Constituents are also called **phrasal nodes** (Klein, Manning, 2003) or, in case of lexicalized parse trees, constituent labels (Collins,1999).
4. **Constituent parsing** is the whole process of generating a constituent tree of a sentence.
5. Hint: Here we only consider nonlexicalized parse trees.

Constituent parsing

There is different possibilities to generate a constituent tree. Some examples are:

1. With a probabilistic context-free grammar (PCFGs)
2. Derived from dependency tree.

Example for a CFG

Internal rules

ROOT	→	S
S	→	NP NP VP
NP	→	JJ NN
NP	→	PRP
NP	→	DT JJ NN
VP	→	VBD NP

Lexical rules

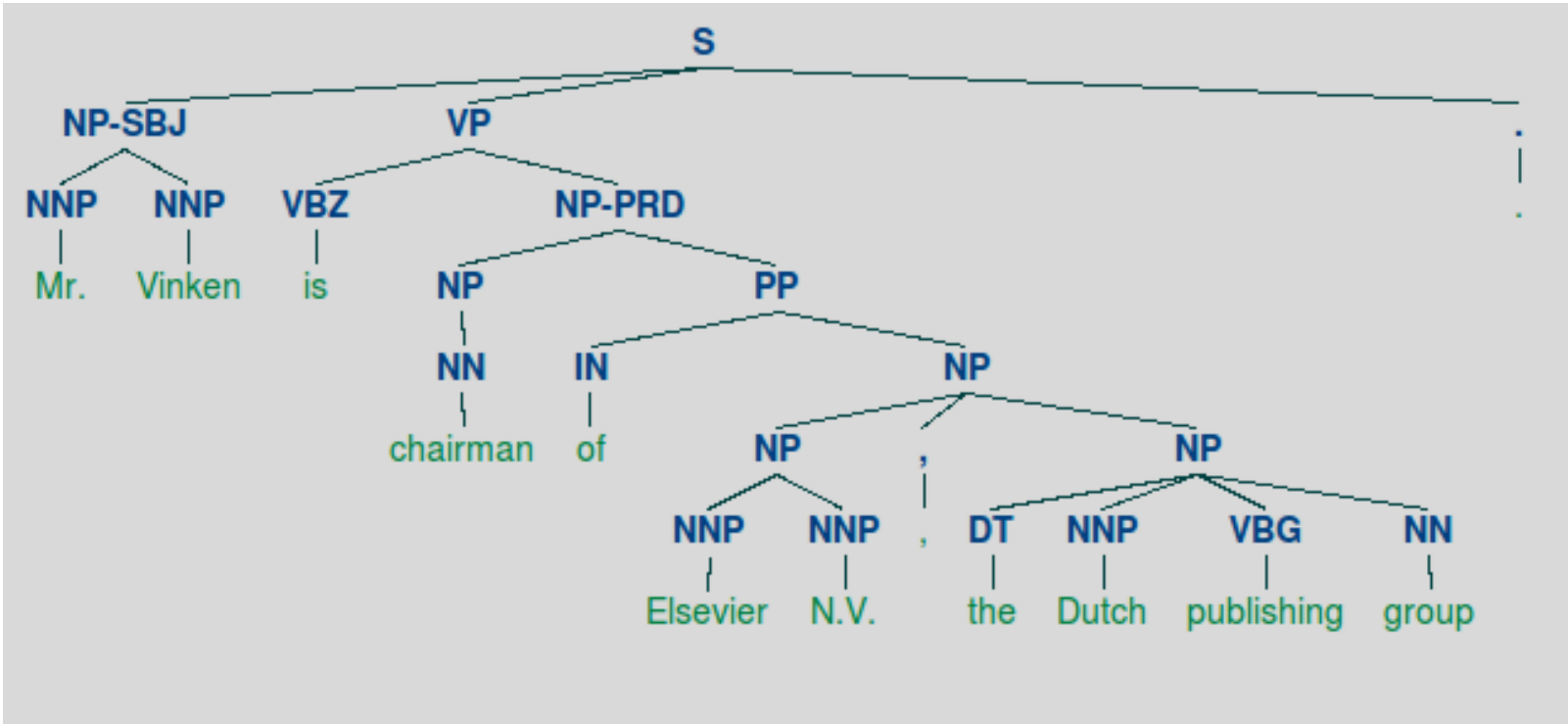
JJ	→	Last | great
NN	→	week | movie
PRP	→	I
VBD	→	saw
DT	→	a

Constituent parsing with PCFG

A constituent tree can be generated by a set of grammatical rules and a probability value for each of it.

1. A **CFG** is a 4-Tuple (N, Σ, A, R), where N is the set of nonterminal symbols (constituents), Σ is an alphabet (where each element $\sigma \in \Sigma$ is a word of our sentence), A is an distinguished start symbol in N (e.g. 'ROOT') and R is a finite set of rules. (Collins,1999)
2. A **PCFG** is CFG, where each rule has a probabilistic value that adds to one for all rules with the same constituent of the LHS (left-hand-site).
3. Rules that only contains nonterminal symbols from N are called **internal rules** and rules that contains terminal symbols on the RHS (right-hand-site) are called **lexical rules**.

Example from the Penn Treebank (provided by the NL-Toolkit)

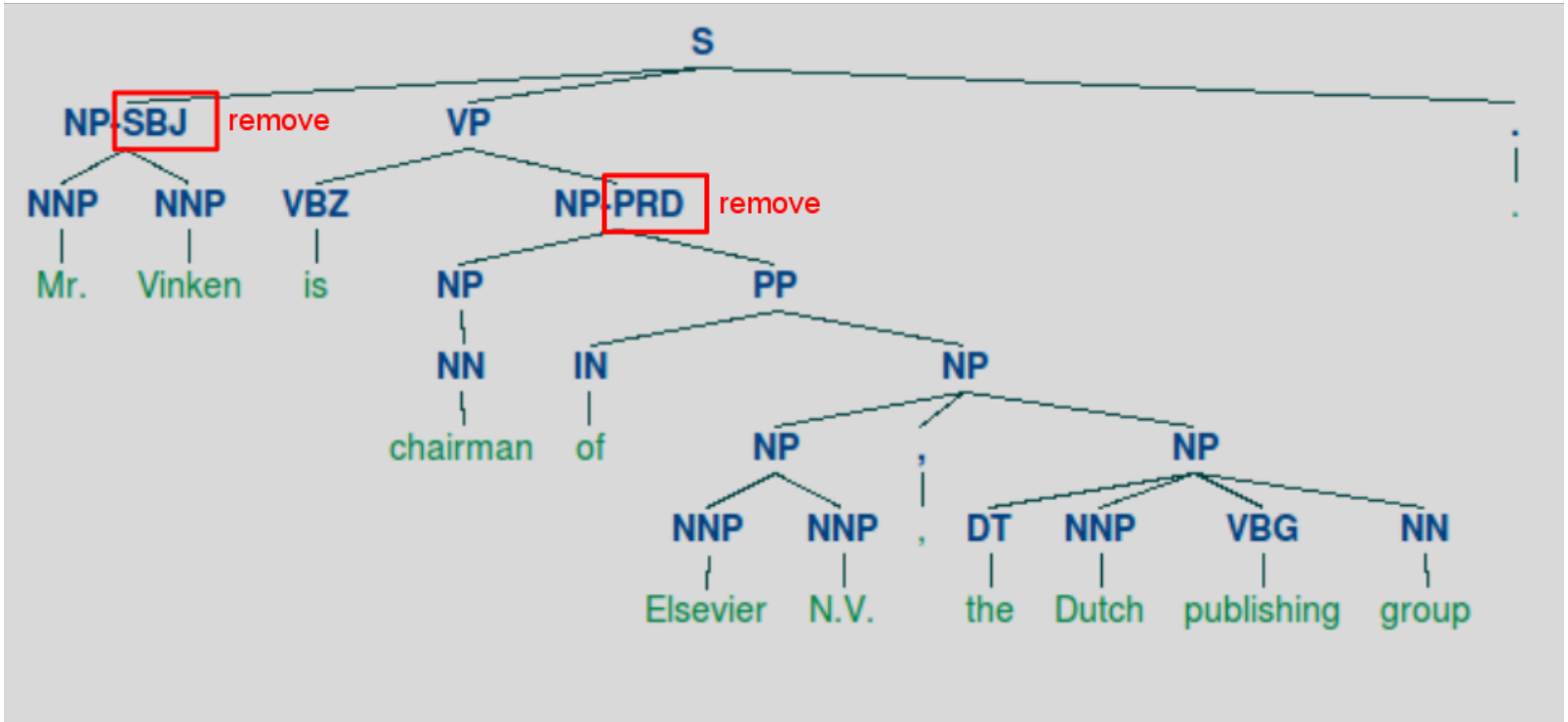


Compute the probability for the PCFG

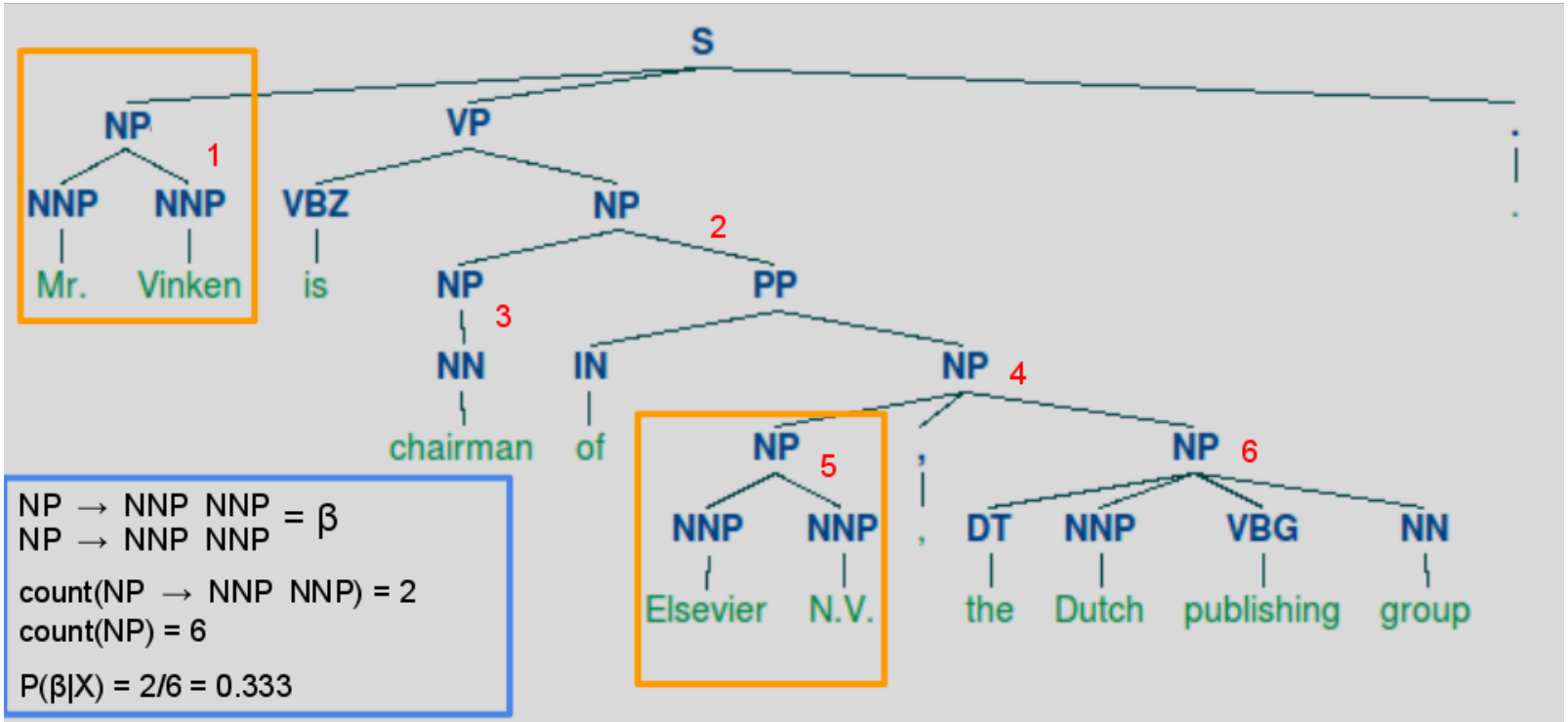
The PCFG can be derived from a corpus like Penn Treebank (that already uses constituents):

1. Take a constituent tree T from the corpus.
2. Simplefy T (e.g. delete functional annotaiton like 'SBJ' and empty nodes).
3. Transform T into a set of internal and lexical rules $r = X \rightarrow \beta$ and add them to R.
4. Start with the next T until all T's were transformed.
5. Calculate $P(\beta|X) = \frac{Count(X \rightarrow \beta)}{Count(X)}$ (Collins, 1999).

Example from the Penn Treebank (provided by the NL-Toolkit)



Example from the Penn Treebank (provided by the NL-Toolkit)



Probability of a constituent tree

1. Given a PCFG and an arbitrary sentence, we can now try to construct a constituent trees.
2. But this trees are not unambigous.
3. Computing a probabilistic value help to decide the right one.
4. The probability of a constituent tree are the product of the n probabilities of the rules that n times applied.
5. Be T the tree, S the sentence, n the num. of applications and $LHS_i \rightarrow RHS_i \in R$. $P(T,S) = \prod_{i=1}^{n} P(RHS_i|LHS_i)$. (Collins, 1999).

Problems with the Penn Treebank

Now, we can generate a PCFG of a corpus like the Penn Treebank, but there with some problems:

1. Very flat rules. (Many variables on the RHS.)
2. Many of the rules are very simular.
3. The VP symbol is very overloaded and there is no decission between finite VP and infinite VP.
4. Many of thes rules exist only at once, so the probabilistic for such results to 1.0

Solutions

- Vertical and horizontal markovization of the rules. (Klein, Manning,2003)
- Seperate symbols for NP used as subject or object. (Subject NP are 8.7 times more expanded to an pronoun than a object NP)
- Head-Annotation.
- Definition of a distanz.
- Detection and labeling of infinit and finit VP.

Vertical and horizontal markovization.

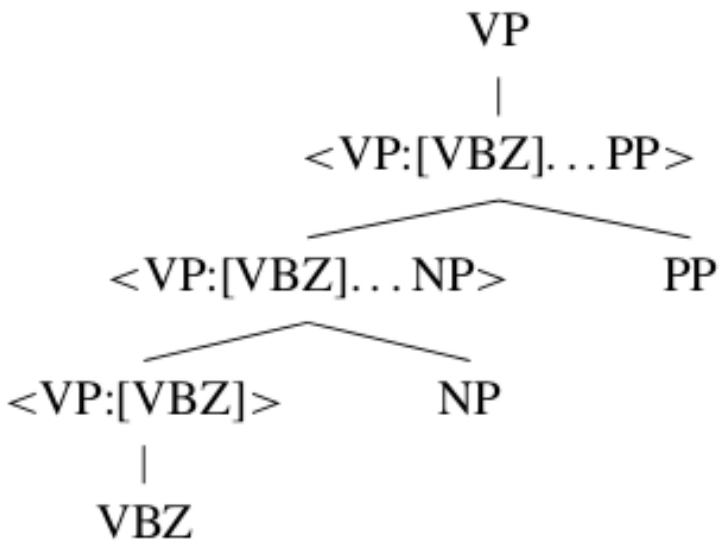


Figure 1: The v=1, h=1 markovization of VP $\rightarrow$ VBZ NP PP.

An other approach.

Create a constituent tree from a given dependency parse.

1. First-Pass Generation: Create a domain structure.
2. Use rules to complement the constituents.
3. Merging the constituents.

Domain	Starting links	Constituent	Comments
p	MVp,Mp,MVt,MX#x,MG,OF,Pp	PP	Prepositional phrase: "The dog ran [PP in the park]."
v	S(except S##d), Pg,Pv,I,PP,PF, SF,SX,Mv,Mg	VP	Verb phrase: "The dog [VP will [VP run in the park]] ."

Software

1. Stanford Parser
2. Link Grammar
3. NLTK

Tests

1. Stanford Parser:
   16 minutes, 33.964 seconds for 42261 words in 1855 sentences.
2. Link Grammar:
   2.65 seconds for its own corpus (900 sentences).

Literature

1. Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430. (Rferenced on the standford parser website)
2. Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, Univ. of Pennsylvania.

Thank you!