Introduction to Natural Language Processing

DATE15/EDA132 – Lecture 3
Parsing Techniques

Pierre Nugues

Pierre.Nugues@cs.lth.se

www.cs.lth.se/~pierre

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Parsing

Constituent parsing (the textbook) is a declining formalism

It cannot properly handle many languages: Swedish, Russian, Czech, Arabic, etc.

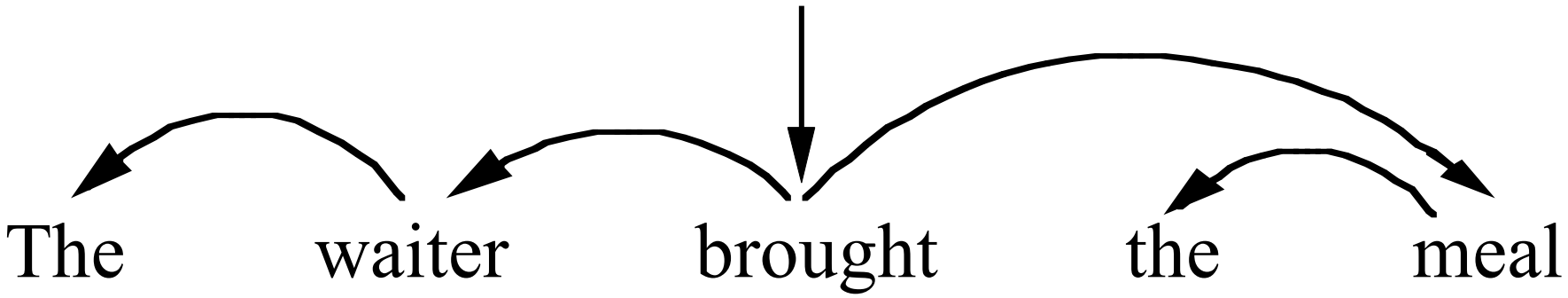
Dependency parsing can handle all these languages as well as English, German, French, etc.

Dependency parsing has improved considerably over the last 4 years: see CONLL 2006 and 2007.

CONLL 2008 and 2009 extend it to semantic parsing

You will implement such a dependency parser for Swedish or another language

A Sentence Tree – Stemma

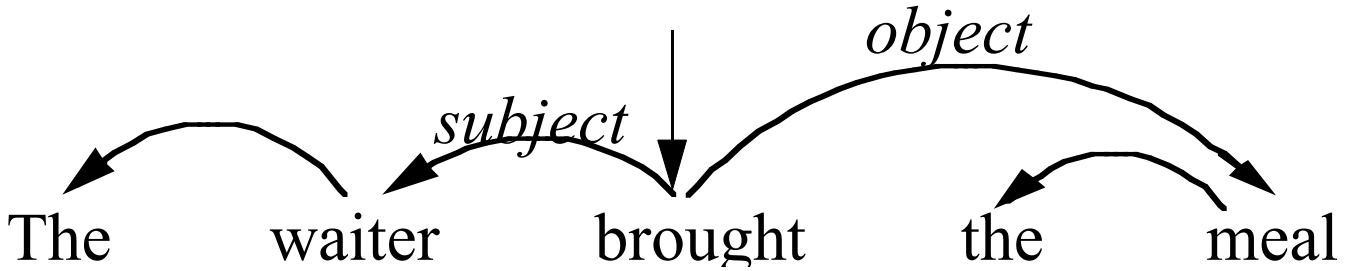


Dependencies and Grammatical Functions

The dependency structure generally reflects the traditional syntactic representation

The links can be annotated with grammatical function labels.

In a simple sentence, it corresponds to the subject and the object



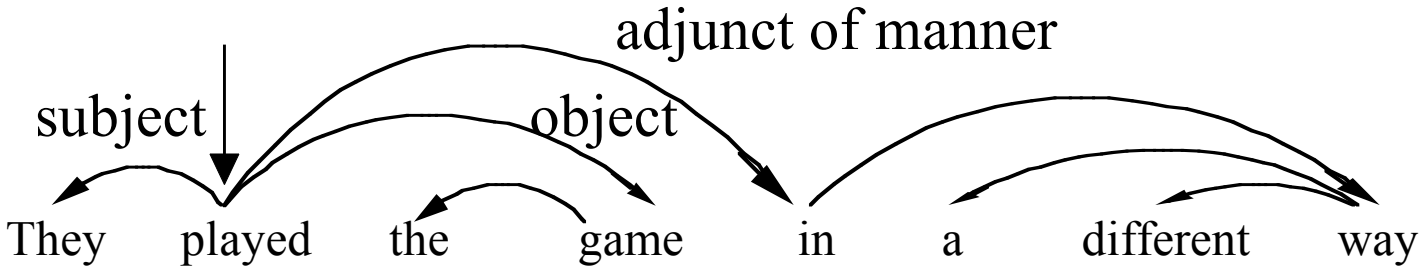
Probably a more natural description to tie syntax to semantics

Dependencies and Functions (II)

Adjuncts form another class of functions that modify the verb

They include prepositional phrases whose head is set arbitrarily to the front preposition

Adjuncts include adverbs that modify a verb



Parser Implementation

The dependency parser takes a sentence as input and produces a dependency graph as output

Ambiguity can yield two or more possible graphs

The parser will use machine-learning techniques to rank parse trees

You will train your parser using a hand-annotated corpus – the training set

You will evaluate it using another set – the test set

Data sets are available from the CONLL 2006 pages: http://nextens.uvt.nl/~conll/

Example of an Annotated Corpus

1	Äktenskapet	_	NN	NN	_	4	SS
2	och	_	++	++	_	3	++
3	familjen	_	NN	NN	_	1	CC
4	är	_	AV	AV	_	0	ROOT
5	en	_	EN	EN	_	7	DT
6	gammal	_	AJ	AJ	_	7	AT
7	institution	_	NN	NN	_	4	SP
8	,	_	IK	IK	_	7	IK
9	som	_	PO	PO	_	10	SS
10	funnits	_	VV	VV	_	7	ET
11	sedan	_	PR	PR	_	10	TA
12	1800-talet	_	NN	NN	_	11	PA
13	.	_	IP	IP	_	4	IP

Parser Input

1	Äktenskapet	_	NN	NN	_
2	och	_	++	++	_
3	familjen	_	NN	NN	_
4	är	_	AV	AV	_
5	en	_	EN	EN	_
6	gammal	_	AJ	AJ	_
7	institution	_	NN	NN	_
8	,	_	IK	IK	_
9	som	_	PO	PO	_
10	funnits	_	VV	VV	_
11	sedan	_	PR	PR	_
12	1800-talet	_	NN	NN	_
13	.	_	IP	IP	_

Part-of-Speech Ambiguity in Swedish

First step of parsing is a part-of-speech tagging.

The word *som* in the *Norstedts svenska ordbok*, 1999. Three entries:

1. *Om jag vore lika vacker som du, skulle jag vara lycklig.* (konjunktion)
2. *Bilen som jag köpte ifjol.* (pronomen)
3. *Som jag har saknat dig.* (adverb)

POS difference is significant:

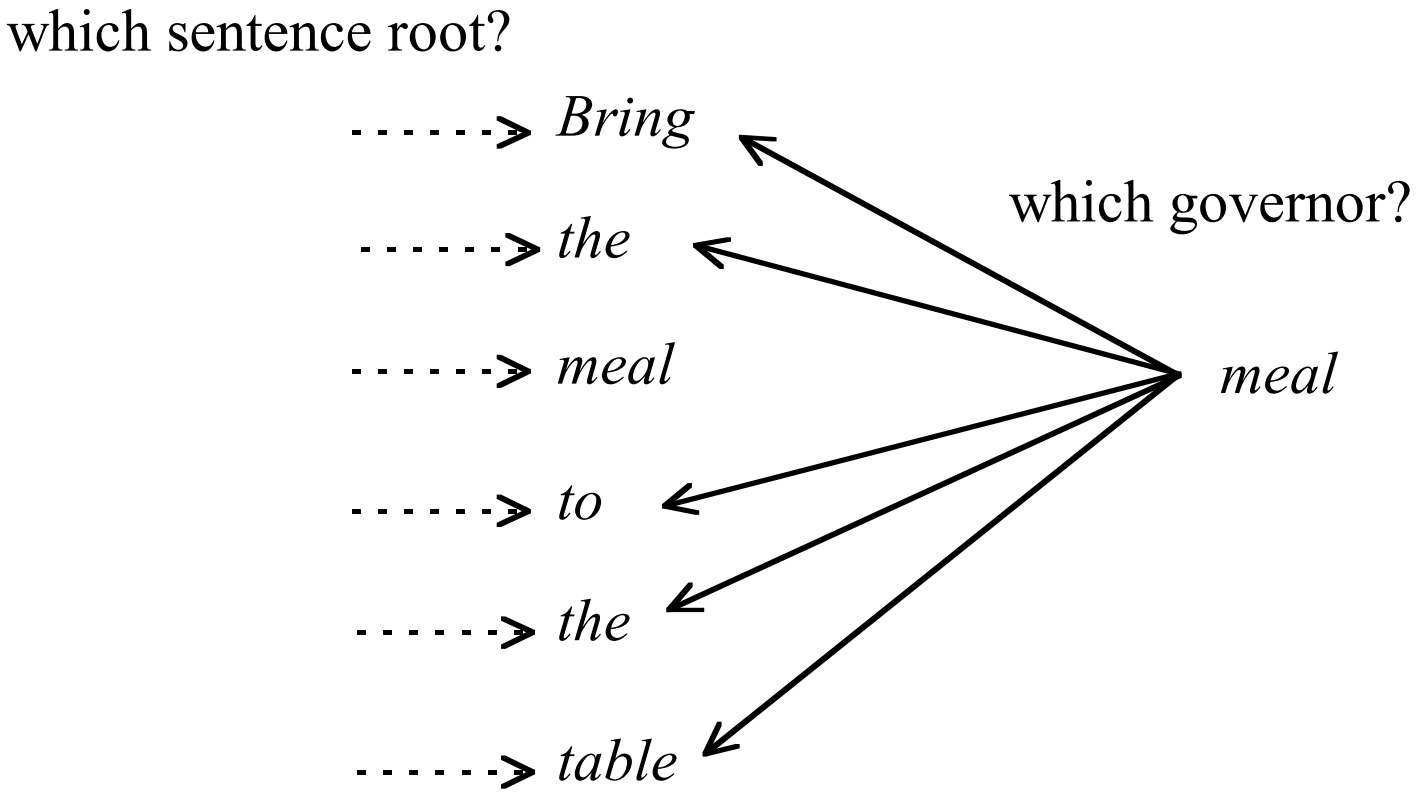
- Compare the pronunciation of *vaken*, adjective and *vaken*, noun in Swedish
- In English, compare *object* in *I object to violence*, verb, or *I could see an object*, noun.

POS Tagging

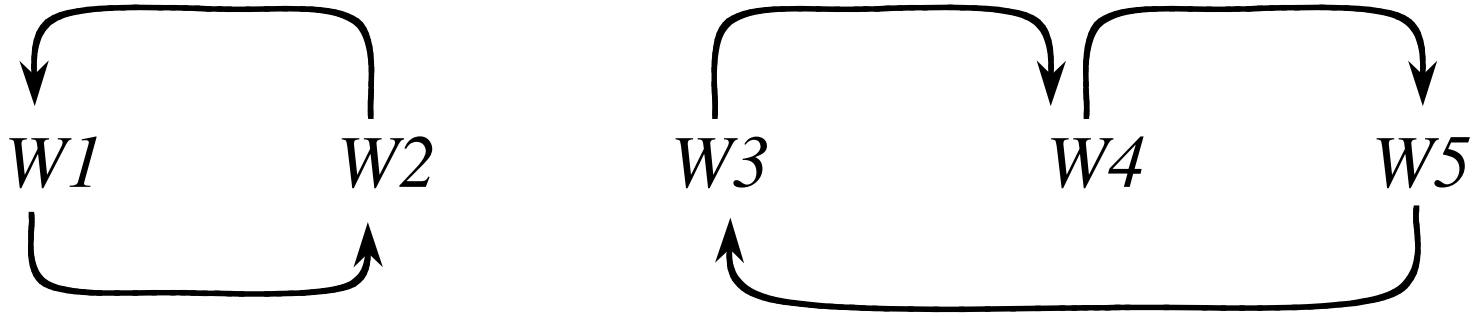
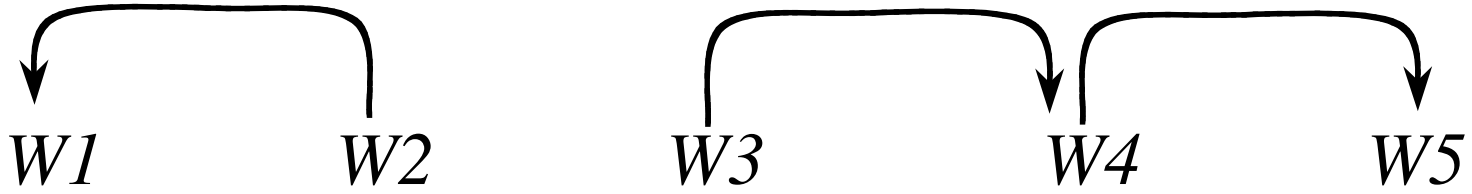
Words	Possible Tags	Example of Use
that	subordinating conjunction	***that he can swim is good***
	determiner	***that white table***
	adverb	***it is not that easy***
	pronoun	***that is the table***
	relative pronoun	***the table that collapsed***
round	verb	***round the usual suspects***
	preposition	***turn round the corner***
	noun	***a big round***
	adjective	***a round box***
	adverb	***he went round***
table	noun	***that white table***
	verb	***I table that***
might	noun	***the might of the wind***
	modal verb	***she might come***
collapse	noun	***the collapse of the empire***
	verb	***the empire can collapse***

Parsing Dependencies

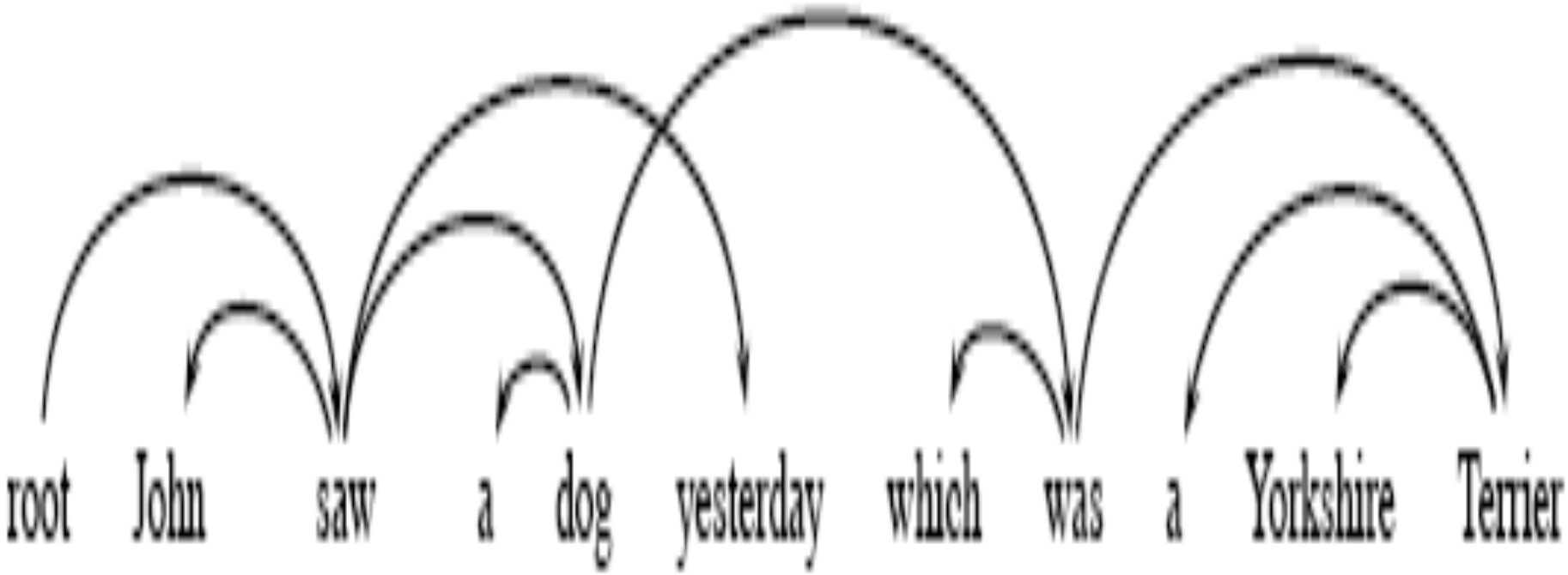
Generate all the pairs:



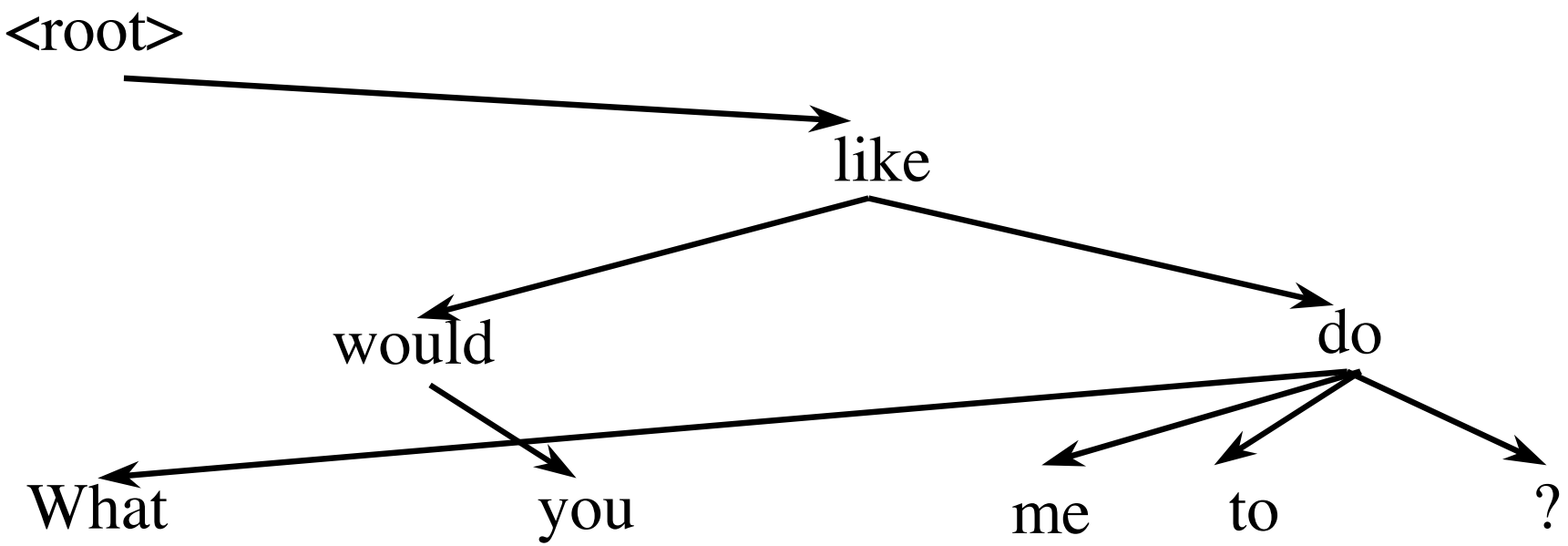
Properties of Dependency Graphs

Acyclic	
Connected	
Projective	Each pair of words (*Dep*, *Head*), directly connected, is only separated by direct or indirect dependents of *Dep* or *Head*

Nonprojective Graphs (McDonald and Pereira)

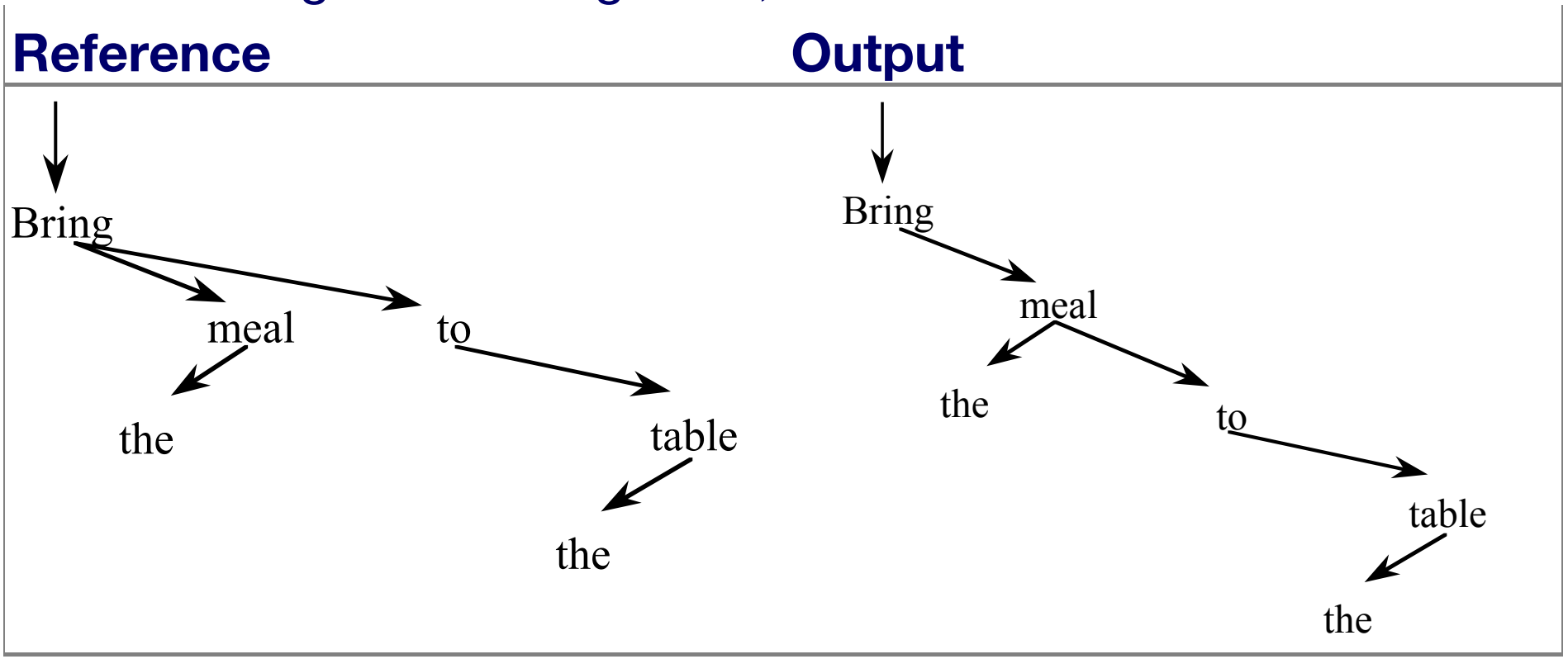


Nonprojective Graphs (Järvinen and Tapanainen)



Parser Evaluation

Dependency parsing: The error count is the number of words that are assigned a wrong head, here 1/6

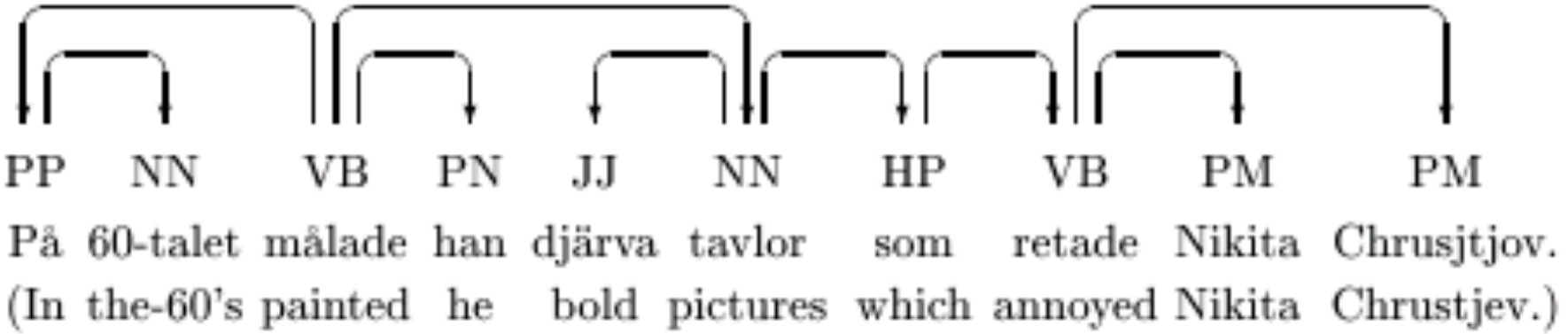


Nivre's Parser for Swedish

Joakim Nivre designed a dependency parser for Swedish.

He reported the best results for this language so far together with a team from Microsoft.

It is the best reported parser for many other languages



Nivre's Parser

The first step is a POS tagging

The parser applies a variation/extension of the shift-reduce algorithm since dependency grammars have no nonterminal symbols

It use four types of actions involving part-of-speech and words

The actions are:

- Shift, pushes the input token to the stack
- Reduce, reduces the token on the top of the stack
- Left arc, adds an arc from the next input token to the token on the top of the stack and reduces it.
- Right arc, adds an arc from the token on top of the stack to the next input token and pushes the input token on the top of the stack.

Actions' Definition

Actions	Definitions $\langle Stack, Queue, Graph \rangle$	Constraints
Init.	$\langle nil, W, \varnothing \rangle$	
Term.	$\langle S, nil, A \rangle$	
Shift	$\langle S, n \mid I, A \rangle \rightarrow \langle n \mid S, I, A \rangle$	
Reduce	$\langle n \mid S, I, A \rangle \rightarrow \langle S, I, A \rangle$	$\exists n'(n', n) \in A$
Left arc	$\langle n \mid S, n' \mid I, A \rangle \rightarrow \langle S, n' \mid I, A \cup \{(n', n)\} \rangle$	$\neg \exists n''(n'', n) \in A$
Right arc	$\langle n \mid S, n' \mid I, A \rangle \rightarrow \langle n' \mid n \mid S, I, A \cup \{(n, n')\} \rangle$	

Nivre's Parser in Action

$$
\begin{array}{lll}
W = & \begin{array}{ccccc} \text{på} & \text{60-talet} & \text{målade} & \text{han} & \text{tavlor} \\ (\text{in}) & (\text{the-60's}) & (\text{painted}) & (\text{he}) & (\text{pictures}) \end{array} & R \supseteq \{ \begin{array}{l} \text{på} \rightarrow \text{60-talet}, \\ \text{på} \leftarrow \text{målade}, \\ \text{målade} \rightarrow \text{han}, \\ \text{målade} \rightarrow \text{tavlor} \ \} \end{array} \\
 & \langle \textbf{nil}, \text{på 60-talet målade han tavlor}, \emptyset \rangle & \\
\xrightarrow{\textbf{S}} & \langle \text{på}, \text{60-talet målade han tavlor}, \emptyset \rangle & \\
\xrightarrow{\textbf{RA}} & \langle \text{60-talet på}, \text{målade han tavlor}, \{(\text{på}, \text{60-talet})\} \rangle & \\
\xrightarrow{\textbf{R}} & \langle \text{på}, \text{målade han tavlor}, \{(\text{på}, \text{60-talet})\} \rangle & \\
\xrightarrow{\textbf{LA}} & \langle \textbf{nil}, \text{målade han tavlor}, \{(\text{på}, \text{60-talet}), (\text{målade}, \text{på})\} \rangle & \\
\xrightarrow{\textbf{S}} & \langle \text{målade}, \text{han tavlor}, \{(\text{på}, \text{60-talet}), (\text{målade}, \text{på})\} \rangle & \\
\xrightarrow{\textbf{RA}} & \langle \text{han målade}, \text{tavlor}, \{(\text{på}, \text{60-talet}), (\text{målade}, \text{på}), (\text{målade}, \text{han})\} \rangle & \\
\xrightarrow{\textbf{R}} & \langle \text{målade}, \text{tavlor}, \{(\text{på}, \text{60-talet}), (\text{målade}, \text{på}), (\text{målade}, \text{han})\} \rangle & \\
\xrightarrow{\textbf{RA}} & \langle \text{tavlor målade}, \textbf{nil}, \{(\text{på}, \text{60-talet}), (\text{målade}, \text{på}), (\text{målade}, \text{han}), (\text{målade}, \text{tavlor})\} \rangle &
\end{array}
$$

Gold—Standard Parsing

For any projective reference graph in the annotated corpus, it is possible to determine an action sequence – non necessarily unique – that parses the sentence.

The actions are member of the set {**`la`**, **`ra`**, **`re`**, **`sh`**}.

Let TOP be the top of the stack and FIRST, the first token of the input list (the queue), and G the dependency graph.

1. if arc(TOP, FIRST) $\in$ G, then **`ra`**;
2. else if arc(FIRST, TOP) $\in$ G, then **`la`**;
3. else if $\exists$k $\in$ Stack, (arc(FIRST, k) $\in$ G or arc(k, FIRST) $\in$ G), then **`re`**;
4. else **`sh`**.

Parsing a Sentence

When parsing an unknown sentence, we do not know the dependencies yet

However, given a context, for instance the part of speech of the top of the stack and the first in the queue, we can predict an action in the set {**`la`**, **`ra`**, **`re`**, **`sh`**}.

This is precisely what the mechanism the parser will use:

- It extracts features (attributes), for instance (POS_TOP, POS_FIRST)
- It uses them as input to a four-class classifier and determines the next action

Training a Classifier

Gold—standard parsing of a manually annotated corpus produces training data

Step #	Stack	Queue	Action
0.	nil/nil	på/PP	sh
1.	på/PP	60-talet/NN	ra
2.	60-talet/NN	målade/VB	re
3.	nil/nil	målade/VB	la
4.	målade/VB	han/PN	sh
5.	han/PN	tavlor/NN	ra
...	...		

Using Talbanken and CONLL 2006 data, you will train decision trees and implement a parser.

Feature Vectors

You extract one feature (attribute) vector for each parsing action.

The most elementary feature vector consists of two parameters: POS_TOP, POS_FIRST

Nivre et al. (2006) used from 16 to 30 parameters and support vector machines.

As machine-learning algorithm, you can use decision trees, perceptron, or support vector machines.