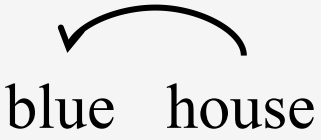
NLP

Introduction to NLP

Dependency Parsing

Dependency Structure



- blue
  - modifier, dependent, child, subordinate
- house
  - head, governor, parent, regent

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.
1 2 3 4 5 6 7 8 9 10

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.

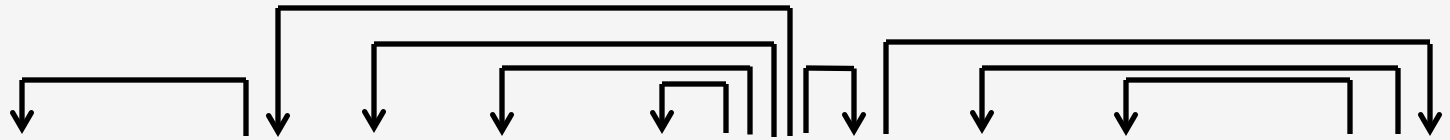
1 2 3 4 5 6 7 8 9 10

Dependency Structure

Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

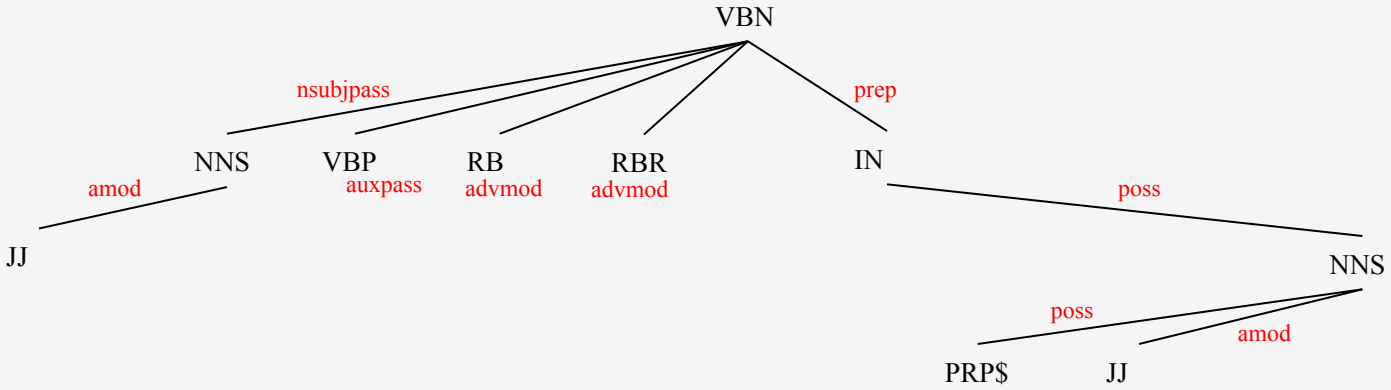
Dependency Structure



Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

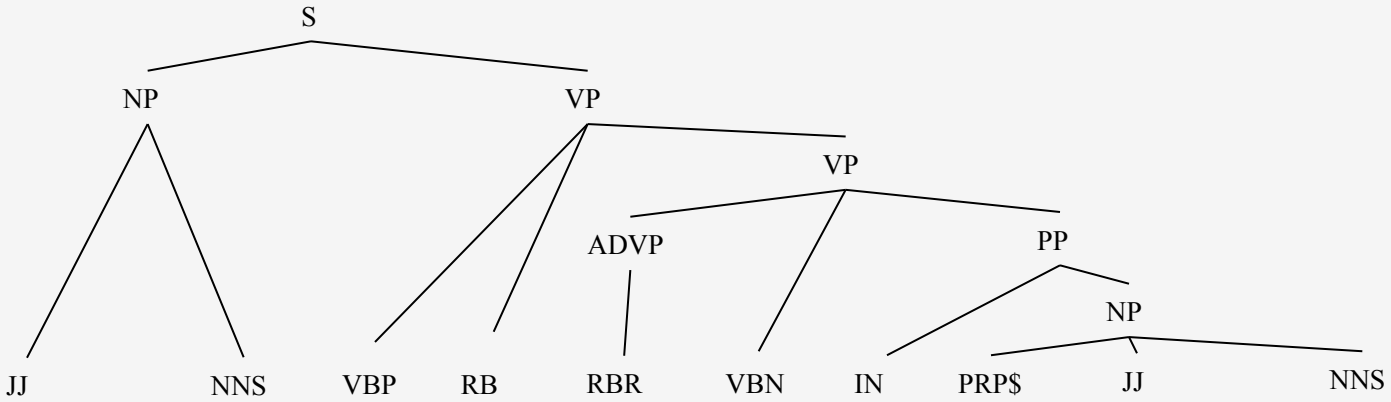
Other Notations



Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

Phrase Structure



Unionized workers are usually better paid than their non-union counterparts.

1 2 3 4 5 6 7 8 9 10

Dependency Grammars

- Characteristics
  - Lexical/syntactic dependencies between words
  - The top-level predicate of a sentence is the root
  - Simpler to parse than context-free grammars
  - Particularly useful for free word order languages

How To Identify The Heads

- H=head, M=modifier
  - H determines the syntactic category of the construct
  - H determines the semantic category of the construct
  - H is required; M may be skipped
  - Fixed linear position of M with respect to H

Head Rules From Collins

Parent Non-terminal	Direction	Priority List
ADJP	Left	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	Right	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	Right	CC RB IN
FRAG	Right	
INTJ	Left	
LST	Right	LS :
NAC	Left	NN NNS NNP NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
PP	Right	IN TO VBG VBN RP FW
PRN	Left	
PRT	Right	RP
QP	Left	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	Right	VP NP ADVP ADJP PP
S	Left	TO IN VP S SBAR ADJP UCP NP
SBAR	Left	WHNP WHPP WHADVP WHADJP IN DT S SQ SINV SBAR FRAG
SBARQ	Left	SQ S SINV SBARQ FRAG
SINV	Left	VBZ VBD VBP VB MD VP S SINV ADJP NP
SQ	Left	VBZ VBD VBP VB MD VP SQ
UCP	Right	
VP	Left	TO VBD VBN MD VBZ VB VBG VBP VP ADJP NN NNS NP
WHADJP	Left	CC WRB JJ ADJP
WHADVP	Right	CC WRB
WHNP	Left	WDT WP WP$ WHADJP WHPP WHNP
WHPP	Right	IN TO FW

Table A.1: The head-rules used by the parser. *Parent* is the non-terminal on the left-hand-side of a rule. *Direction* specifies whether search starts from the left or right end of the rule. *Priority* gives a priority ranking, with priority decreasing when moving down the list.

Techniques (1)

- Dynamic programming
  - CKY – similar to lexicalized PCFG, cubic complexity (Eisner 96)

Techniques (2)

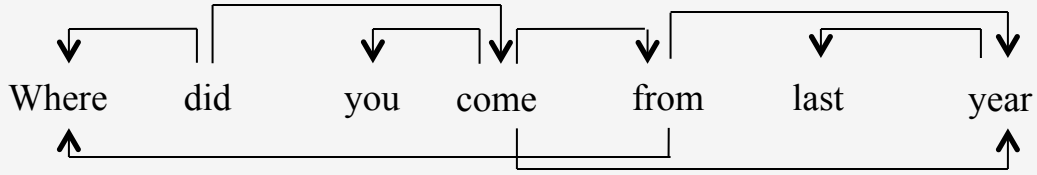
- Constraint-based methods
  - Maruyama 1990, Karlsson 1990
  - Example
    - word(pos(x)) = DET ⇒(label(X) = NMOD, word(mod(x)) = NN, pos(x) < mod(x))
    - A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
  - NP complete problem; heuristics needed
- Constraint graph
  - For initial constraint graph using a core grammar: nodes, domains, constraints
  - Find an assignment that doesn't contradict any constraints. If more than one assignment exists, add more constraints.

Techniques (3)

- Deterministic parsing
  - Covington 2001
  - MaltParser by Nivre
    - shift/reduce as in a shift/reduce parser
    - reduce creates dependencies with the head on either the left or the right
- Graph-based methods
  - Maximum spanning trees (MST)
    - MST Parser by McDonald et al.

Non-projectivity

Output of (the non-projective) MSTParser



Output of Stanford parser

advmod(come-4, Where-1)
aux(come-4, did-2)
nsubj(come-4, you-3)
root(ROOT-0, come-4)
prep(come-4, from-5)
amod(year-7, last-6)
pobj(from-5, year-7)

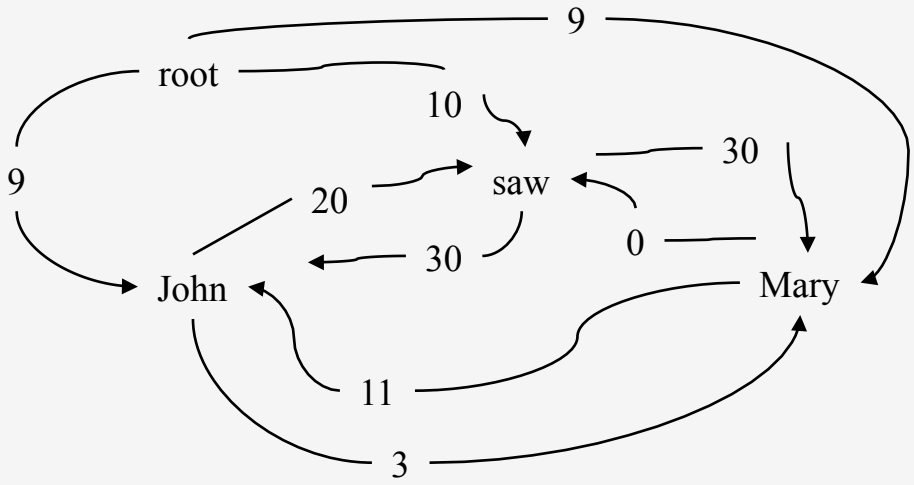
1	Where	Where	WRB	WRB	-	2	SBJ	-	-
2	did	did	VBD	VBD	-	0	ROOT	-	-
3	you	you	PRP	PRP	-	4	SBJ	-	-
4	come	come	VBP	VBP	-	2	VC	-	-
5	from	from	IN	IN	-	4	DIR	-	-
6	last	last	JJ	JJ	-	7	NMOD	-	-
7	year	year	NN	NN	-	5	PMOD	-	-
8	?	?	.	.	-	2	P	-	-

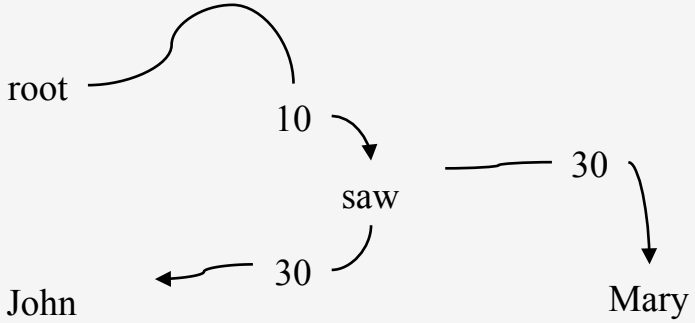
Dependency Parsing

- Background
  - McDonald et al. 2005
- Projectivity
  - English dependency trees are mostly projective (can be drawn without crossing dependencies).
  - Other languages are not.
- Idea
  - Dependency parsing is equivalent to search for a maximum spanning tree in a directed graph.
  - Chu and Liu (1965) and Edmonds (1967) give an efficient algorithm for finding MST for directed graphs.

MST Parser Example

- Consider the sentence “John saw Mary”
- The Chu-Liu-Edmonds algorithm gives the MST on the right hand side (right). This is in general a non-projective tree.





MaltParser (Nivre 2008)

- Very similar to shift-reduce parsing.
- It includes the following components
  - A stack
  - A buffer
  - Set of dependencies (arcs)
- The reduce operations combine an element from the stack and one from the buffer
- Arc-eager parser
  - The actions are shift, reduce, left-arc, right-arc

MaltParser Actions

$$\text{Shift} \quad \frac{[\ldots]_S \quad [w_i, \ldots]_Q}{[\ldots, w_i]_S \quad [\ldots]_Q}$$

$$\text{Reduce} \quad \frac{[\ldots, w_i]_S \quad [\ldots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [\ldots]_Q}$$

$$\text{Left-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_i}{[\ldots]_S \quad [w_j, \ldots]_Q \quad w_i \overset{r}{\leftarrow} w_j}$$

$$\text{Right-Arc}_r \quad \frac{[\ldots, w_i]_S \quad [w_j, \ldots]_Q \quad \neg\exists w_k : w_k \rightarrow w_j}{[\ldots, w_i, w_j]_S \quad [\ldots]_Q \quad w_i \overset{r}{\rightarrow} w_j}$$

[Example from Nivre and Kuebler]

Example

- People want to be free
  - [ROOT] [People, want, to, be, free] Ø
  - Shift [ROOT, People] [want, to, be, free]
  - LA_{nsubj} [ROOT] [want, to, be, free] $A_1 = \{nsubj(want, people)\}$
  - RA_{root} [ROOT, want] [to, be, free] $A_2 = A_1 \cup \{root(ROOT, want)\}$
- The next action is chosen using a classifier
- There is no search
- The final list of arcs is returned as the dependency tree
- Very fast method

Evaluation Metric

- Labeled dependency accuracy
- # correct deps/# deps

1	Unionized	Unionized	VBN	VBN	-	2	NMOD	-	-
2	workers	workers	NNS	NNS	-	3	SBJ	-	-
3	are	are	VBP	VBP	-	0	ROOT	-	-
4	usually	usually	RB	RB	-	3	TMP	-	-
5	better	better	RBR	RBR	-	4	ADV	-	-
6	paid	paid	VBN	VBN	-	5	AMOD	-	-
7	than	than	IN	IN	-	5	AMOD	-	-
8	their	their	PRP$	PRP$	-	10	NMOD	-	-
9	non-union	non-union	JJ	JJ	-	10	NMOD	-	-
10	counterparts	counterparts	NNS	NNS	-	7	PMOD	-	-

Complexity

- Projective (CKY) $O(n^5)$
- Projective (Eisner) $O(n^3)$
- Non-projective (MST – Chu-Liu-Edmonds) $O(n^2)$
- Projective (Malt) $O(n)$

Use In Information Extraction

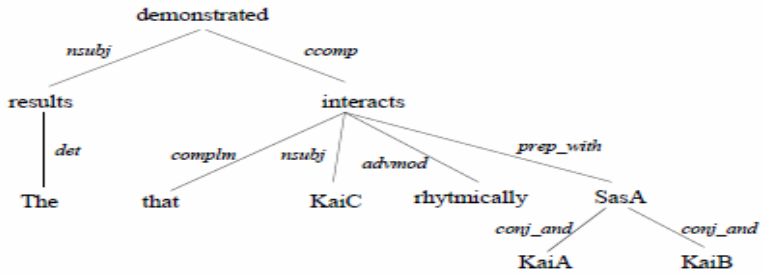


Figure 1: The dependency tree of the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA.*"

1. KaiC - nsubj - interacts - prep_with - SasA
2. KaiC - nsubj - interacts - prep_with - SasA - conj_and - KaiA
3. KaiC - nsubj - interacts - prep_with – SasA - conj_and - KaiB
4. SasA - conj_and - KaiA
5. SasA - conj_and - KaiB
6. KaiA – conj_and – SasA - conj_and - KaiB

1. *PROTX1* - nsubj - interacts - prep_with - *PROTX2*
2. *PROTX1* - nsubj - interacts - prep_with - *PROTX0* - conj_and - *PROTX2*
3. *PROTX1* - nsubj - interacts - prep_with – *PROTX0* - conj_and - *PROTX2*
4. *PROTX1* - conj_and - *PROTX2*
5. *PROTX1* - conj_and - *PROTX2*
6. *PROTX1* – conj_and – *PROTX0* - conj_and - *PROTX2*

[Erkan et al. 2007]

Dependency Kernels

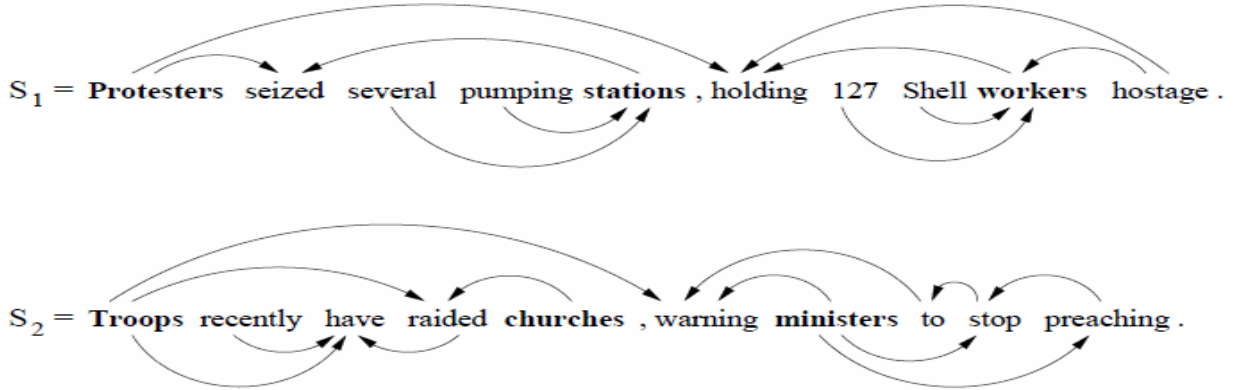


Figure 1: Sentences as dependency graphs.

Relation Instance	Shortest Path in Undirected Dependency Graph
S_1: protesters AT stations	**protesters** ⟶ seized ⟵ **stations**
S_1: workers AT stations	**workers** ⟶ holding ⟵ protesters ⟶ seized ⟵ **stations**
S_2: troops AT churches	**troops** ⟶ raided ⟵ **churches**
S_2: ministers AT churches	**ministers** ⟶ warning ⟵ troops ⟶ raided ⟵ **churches**

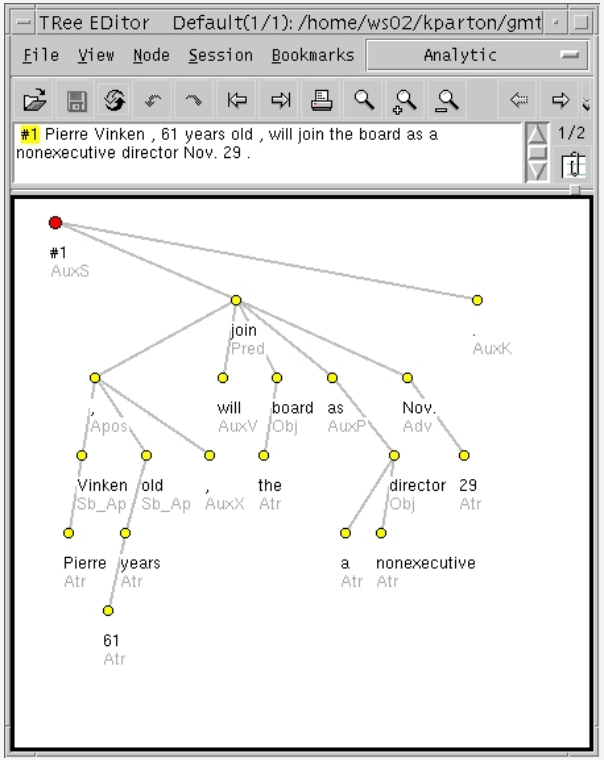
Table 1: Shortest Path representation of relations.

[Bunescu and Mooney 2005]

External Links

- http://ilk.uvt.nl/conll/
  - CONLL-X Shared task
- http://ufal.mff.cuni.cz/pdt2.0/
  - Prague Dependency Treebank
- http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite
- http://nextens.uvt.nl/depparse-wiki/DataOverview
- http://maltparser.org/
  - Joakim Nivre's Maltparser
- http://www.cs.ualberta.ca/~lindek/minipar.htm
  - Dekang Lin's Minipar
- http://www.link.cs.cmu.edu/link/
  - Daniel Sleator and Davy Temperley's Link parser

Prague Dependency Treebank Example



NLP