Computational Linguistics

2B

CSC 485/2501
Fall 2022

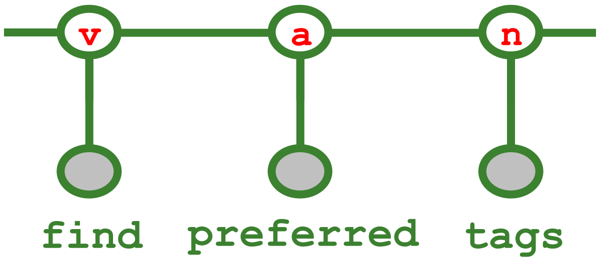
2B. Graphical Dependency Parsing

Gerald Penn
Department of Computer Science, University of Toronto

Based on slides by Yuji Matsumoto, Dragomir Radev, David Smith, Sam Thomson and Jason Eisner

Predicting structured outputs

- Log-linear models great for n-way classification
- Also good for predicting sequences



CVEs, or, to allow fast dynamic programming, only use **n-gram** features

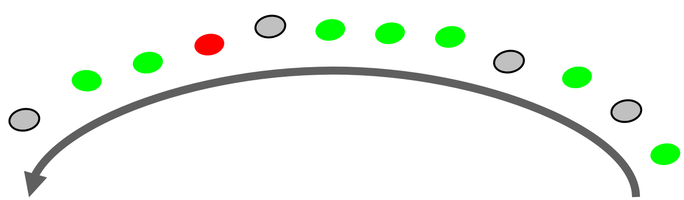
- Also good for dependency parsing

but to allow fast dynamic programming or MST parsing, only use **single-edge** features

Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

yes, lots of green ...



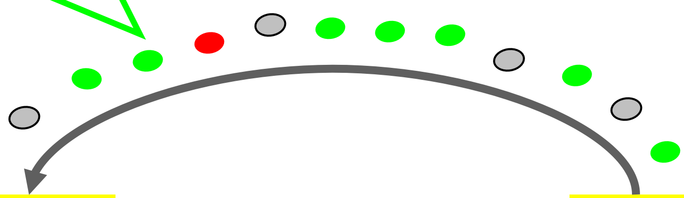
Byl jasný studený dubnový den a hodiny odbíjely třináctou

“It was a bright cold day in April and the clocks were striking thirteen”

Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

jasný ← den
("bright day")

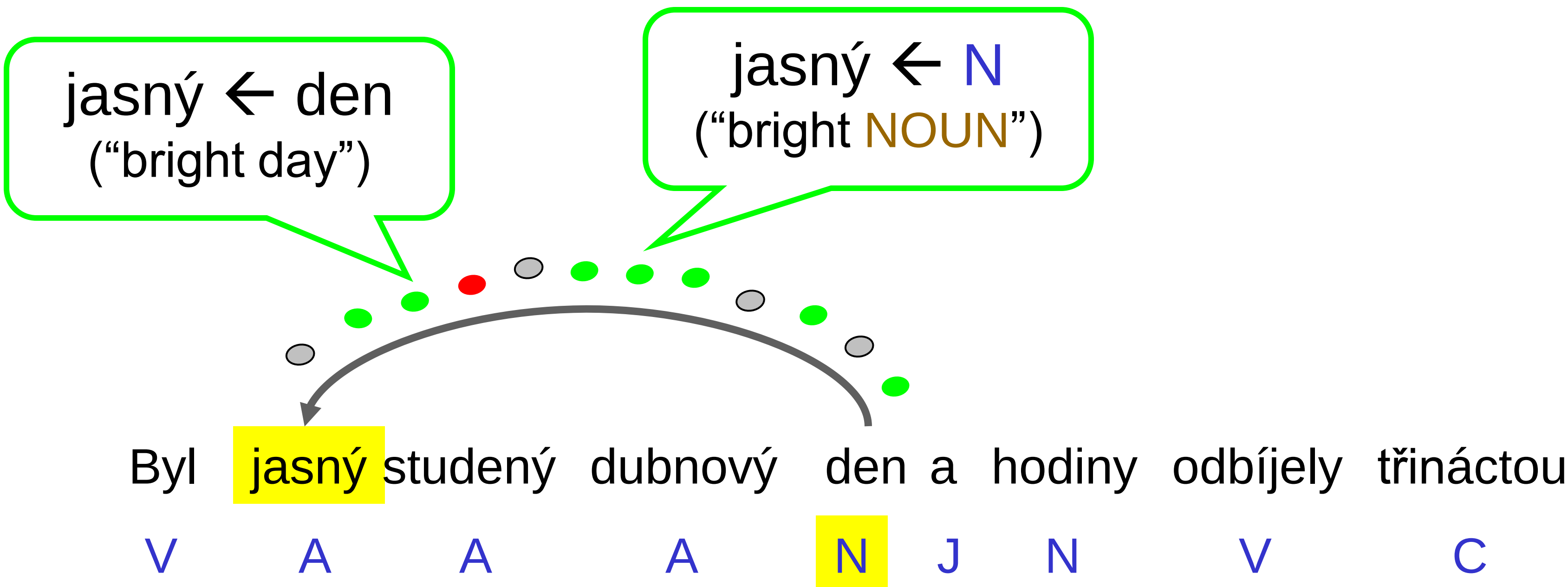


Byl jasný studený dubnový den a hodiny odbíjely třináctou

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

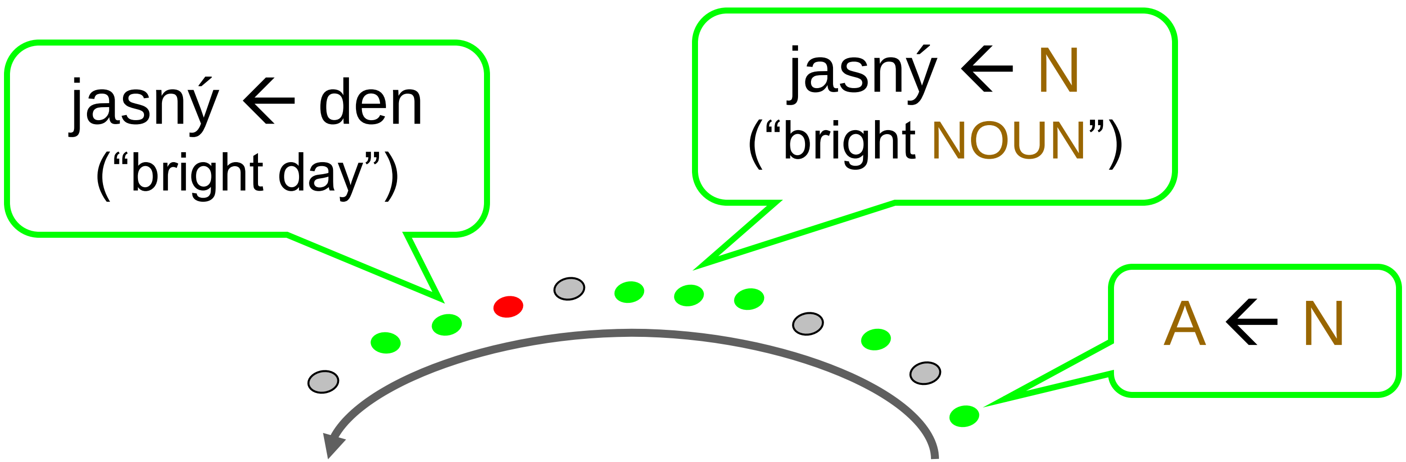
- Is this a good edge?



"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

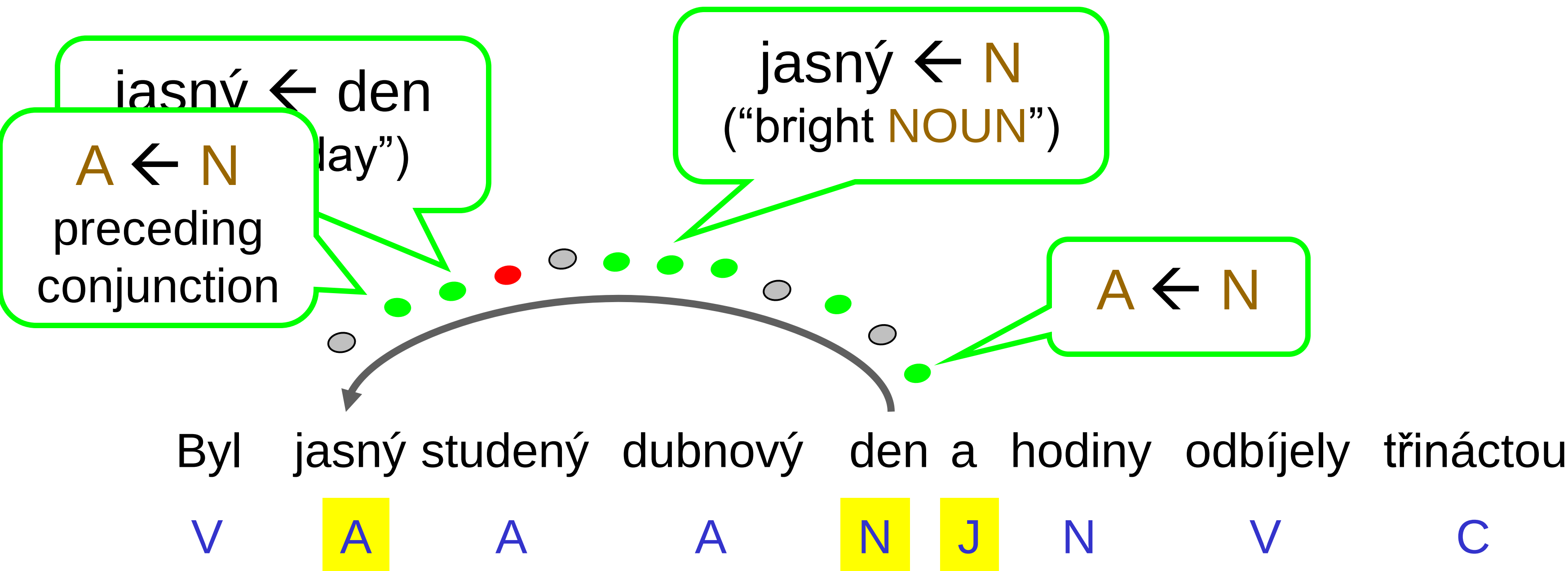


Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

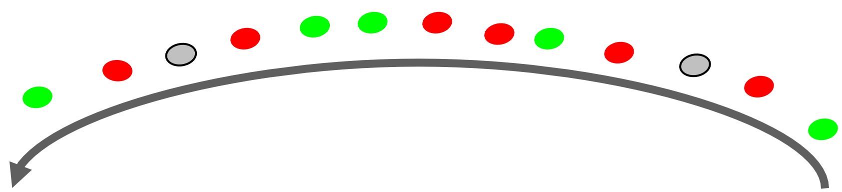


“It was a bright cold day in April and the clocks were striking thirteen”

Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?

not as good, lots of red ...



Byl jasný studený dubnový den a hodiny odbíjely třináctou

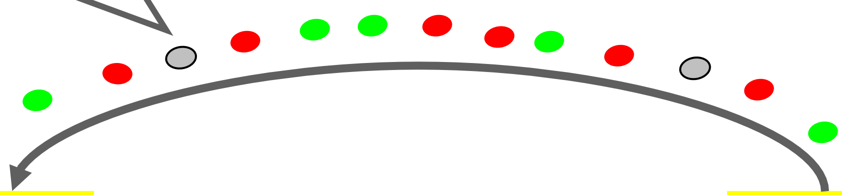
V A A A N J N V C

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?

jasný ← hodiny
(“bright clocks”)
... undertrained ...

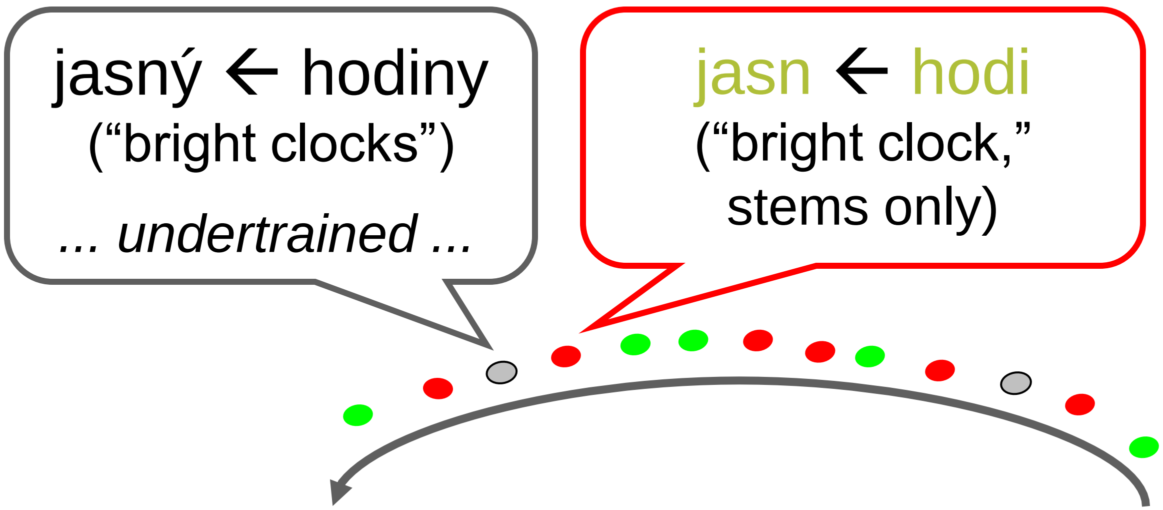


Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C

“It was a bright cold day in April and the clocks were striking thirteen”

Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?



Byl jasný studený dubnový den a hodiny odbíjely třináctou

V A A A N J N V C

byl jasn stud dubn den a hodi odbí třin

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?

jasný ← hodiny
("bright clocks")
... undertrained ...

jasn ← hodi
("bright clock," stems only)

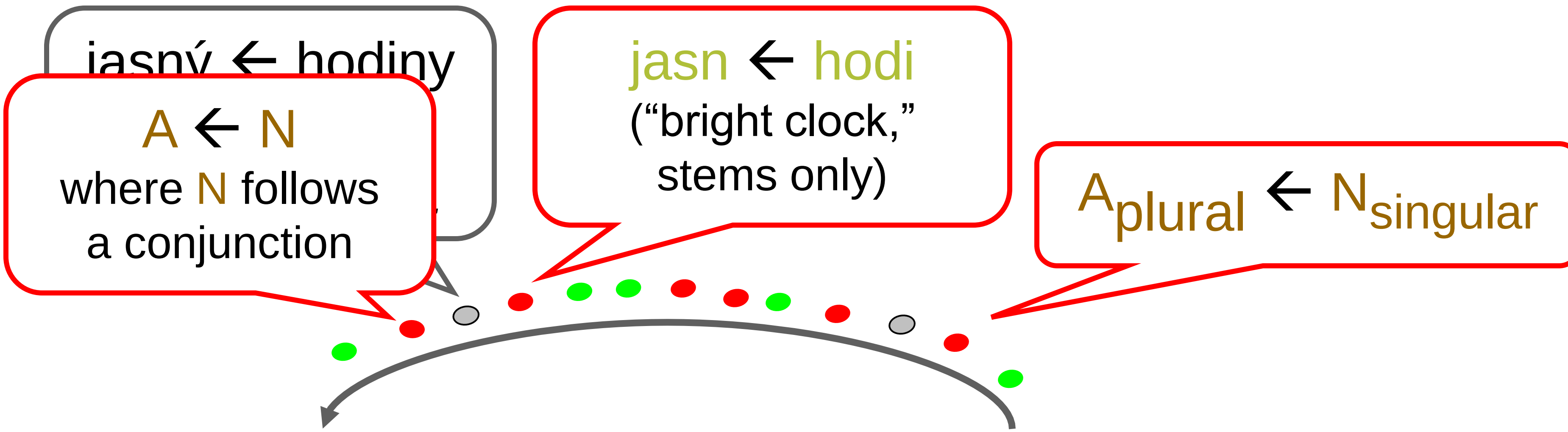
$A_{plural} \leftarrow N_{singular}$

Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?

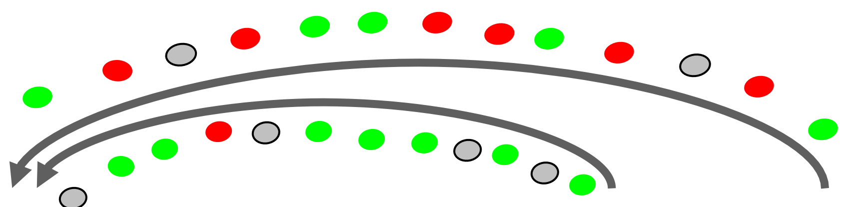


Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
  - “bright day” or “bright clocks”?



Byl jasný studený dubnový den a hodiny odbíjely třináctou

V A A A N J N V C

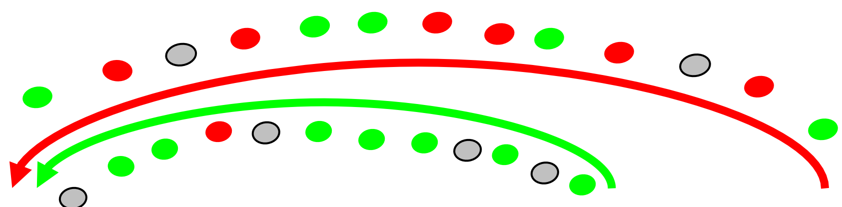
byl jasn stud dubn den a hodi odbí třin

“It was a bright cold day in April and the clocks were striking thirteen”

Edge-Factored Parsers (McDonald et al. 2005)

our current weight vector

- Which edge is better?
- Score of an edge e = θ · **features**(e)
- Standard algos ➔ valid parse with max <u>total</u> score



Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

“It was a bright cold day in April and the clocks were striking thirteen”

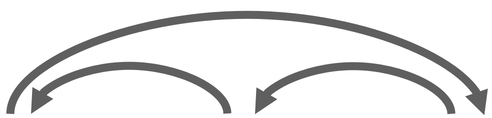
Edge-Factored Parsers (McDonald et al. 2005)

our current weight vector

- Which edge is better?
- Score of an edge e = $\theta \cdot$ **features**(e)
- Standard algos ➔ valid parse with max <u>total</u> score

can't have both
(one parent per word)

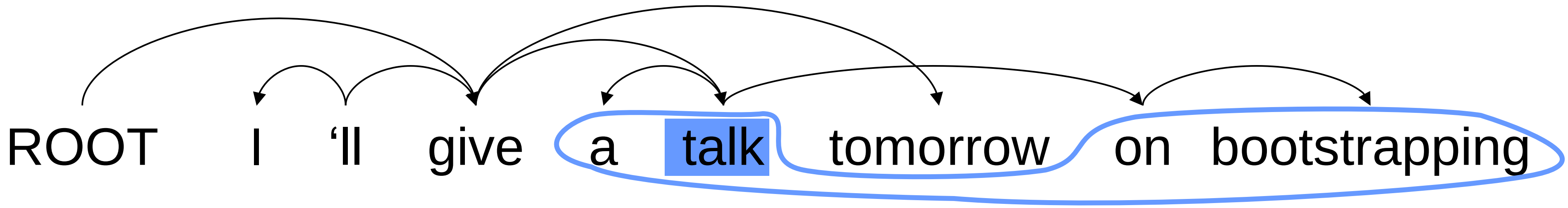
can't have both
(no crossing links)



Can't have all three
(no cycles)

Thus, an edge may lose (or win) because of a consensus of **<u>other</u>** edges.

Non-Projective Parses

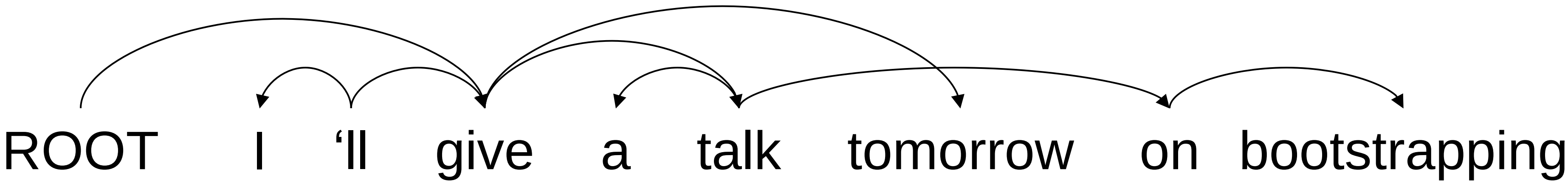


subtree rooted at "talk"
is a **discontiguous** noun phrase

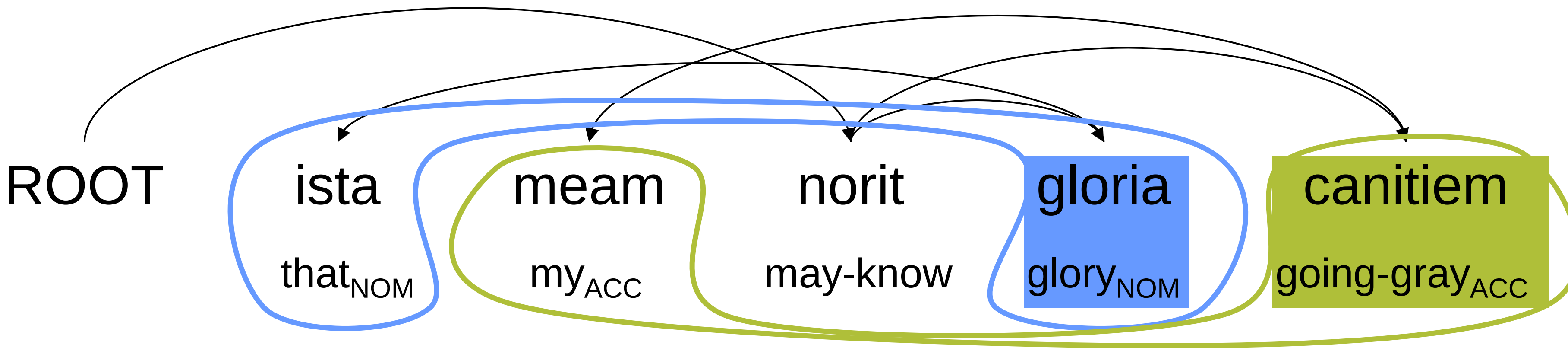
can't have both
(no crossing links)

The "projectivity" restriction.
Do we really want it?

Non-Projective Parses



occasional non-projectivity in English



That glory may-know my going-gray

(i.e., it shall last till I go gray)

frequent non-projectivity in Latin, etc.

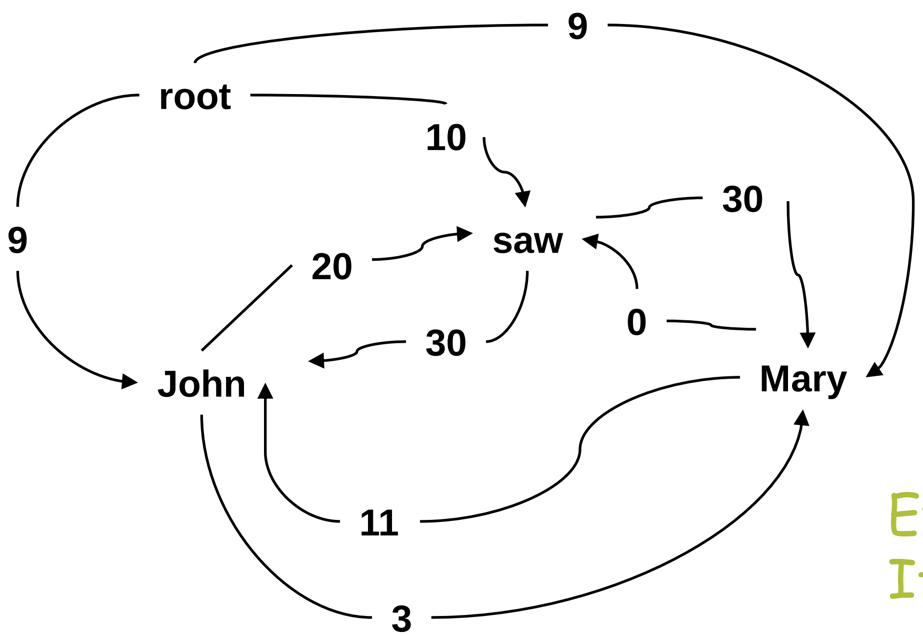
Non-Projective Parsing Algorithms

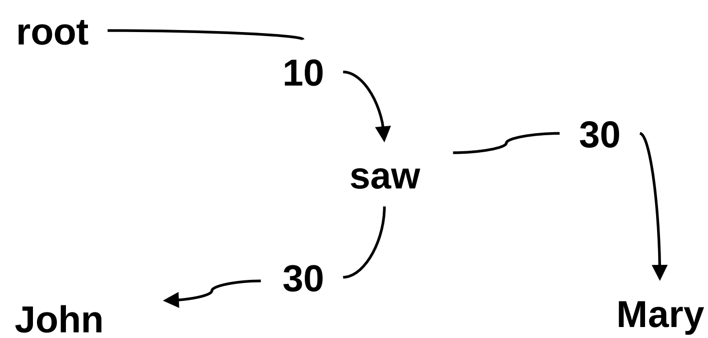
- Complexity considerations:
  - Projective (Proj)
  - Non-projective (NonP)

Problem/Algorithm	Proj	NonP
Complete grammar parsing [Gaifman 1965, Neuhaus and Bröker 1997]	P	NP hard
Deterministic parsing [Nivre 2003, Covington 2001]	$O(n)$	$O(n^2)$
First order spanning tree [McDonald et al. 2005b]	$O(n^3)$	$O(n^2)$
Nth order spanning tree ($N > 1$) [McDonald and Pereira 2006]	P	NP hard

McDonald’s Approach (non-projective)

- Consider the sentence “John saw Mary” (left).
- The Chu-Liu-Edmonds algorithm finds the maximum-weight spanning tree (right) – may be non-projective.
- Can be found in time $O(n^2)$.





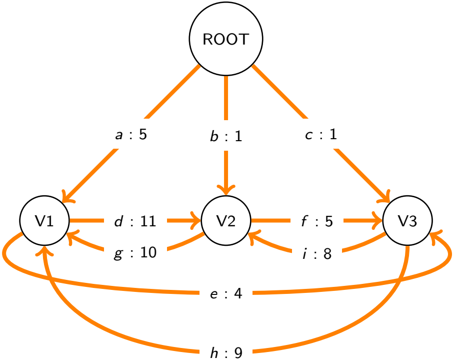
Every node selects best parent
If cycles, contract them and repeat

slide thanks to Dragomir Radev

Chu-Liu-Edmonds - Contracting Stage

- For each non-ROOT node v, set bestInEdge[v] to be its highest scoring incoming edge.
- If a cycle C is formed:
  - contract the nodes in C into a new node v_C
  - edges outgoing from any node in C now get source v_C
  - edges incoming to any node in C now get destination v_C
  - For each node u in C, and for each edge e incoming to u from outside of C:
    - add to e.kicksOut the edge bestInEdge[u], and
    - set e.score to be e.score $-$ e.kicksOut.score.
- Repeat until every non-ROOT node has an incoming edge and no cycles are formed

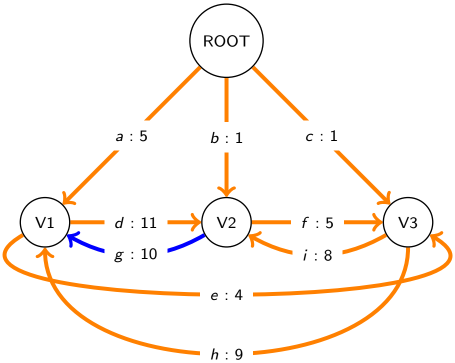
An Example - Contracting Stage



	bestInEdge
V1	
V2	
V3	

	kicksOut
a	
b	
c	
d	
e	
f	
g	
h	
i	

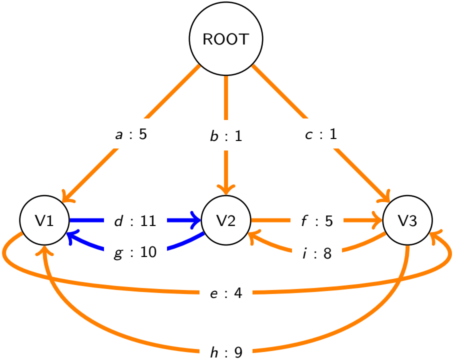
An Example - Contracting Stage



	bestInEdge
V1	g
V2	
V3	

	kicksOut
a	
b	
c	
d	
e	
f	
g	
h	
i	

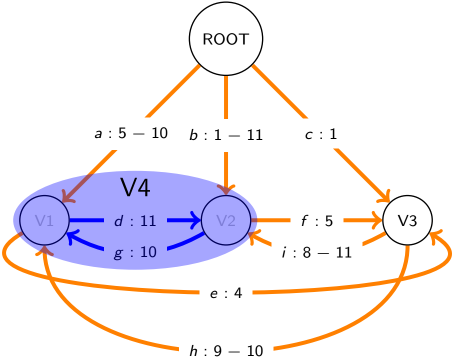
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	

	kicksOut
a	
b	
c	
d	
e	
f	
g	
h	
i	

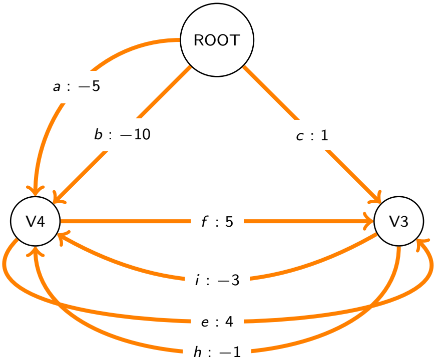
An Example - Contracting Stage



	`bestInEdge`
V1	g
V2	d
V3	

	`kicksOut`
a	g
b	d
c	
d	
e	
f	
g	
h	g
i	d

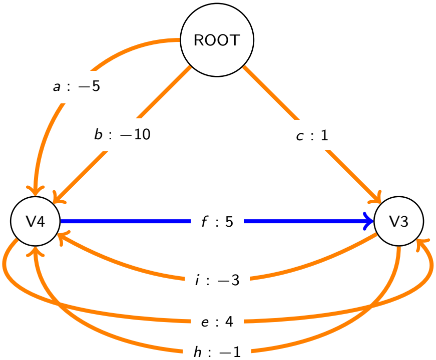
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	
V4	

	kicksOut
a	g
b	d
c	
d	
e	
f	
g	
h	g
i	d

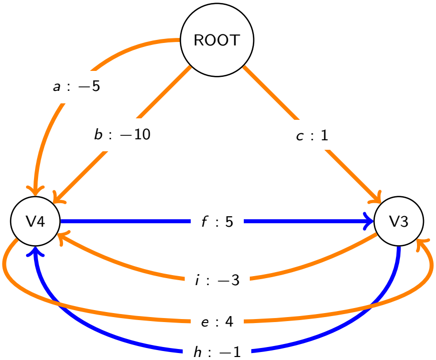
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	f
V4	

	kicksOut
a	g
b	d
c	
d	
e	
f	
g	
h	g
i	d

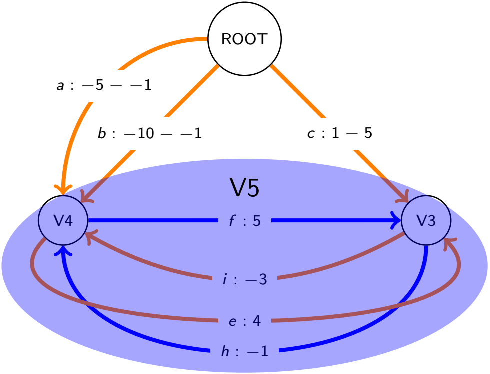
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	f
V4	h

	kicksOut
a	g
b	d
c	
d	
e	
f	
g	
h	g
i	d

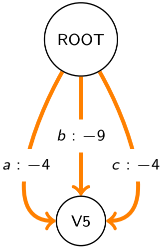
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	f
V4	h
V5	

	kicksOut
a	g, h
b	d, h
c	f
d	
e	
f	
g	
h	g
i	d

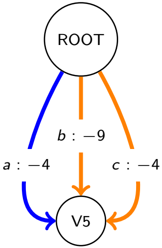
An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	f
V4	h
V5	

	kicksOut
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

An Example - Contracting Stage



	bestInEdge
V1	g
V2	d
V3	f
V4	h
V5	a

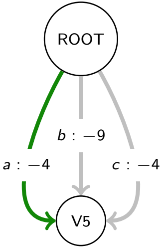
	kicksOut
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

Chu-Liu-Edmonds - Expanding Stage

After the contracting stage, every contracted node will have exactly one `bestInEdge`. This edge will kick out one edge inside the contracted node, breaking the cycle.

- Go through each `bestInEdge` e in the *reverse* order that we added them
- lock down e, and remove every edge in `kicksOut`(e) from `bestInEdge`.

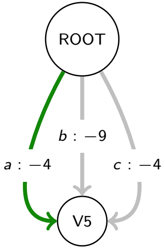
An Example - Expanding Stage



	`bestInEdge`
V1	g
V2	d
V3	f
V4	h
V5	a

	`kicksOut`
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

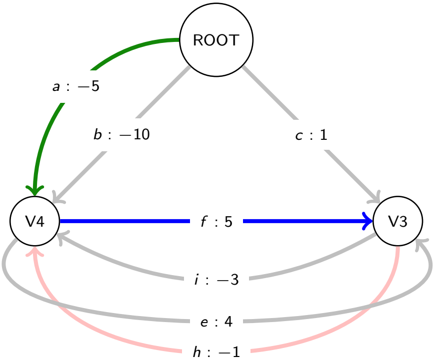
An Example - Expanding Stage



	bestInEdge
V1	a ~~g~~
V2	d
V3	f
V4	a ~~h~~
V5	a

	kicksOut
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

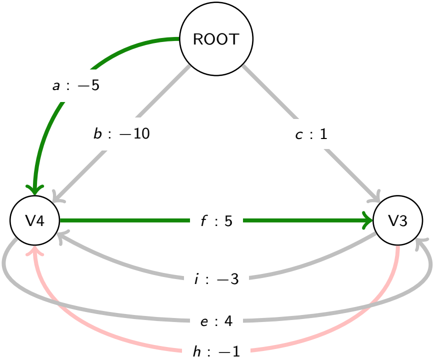
An Example - Expanding Stage



	bestInEdge
V1	a ~~g~~
V2	d
V3	f
V4	a ~~h~~
V5	a

	kicksOut
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

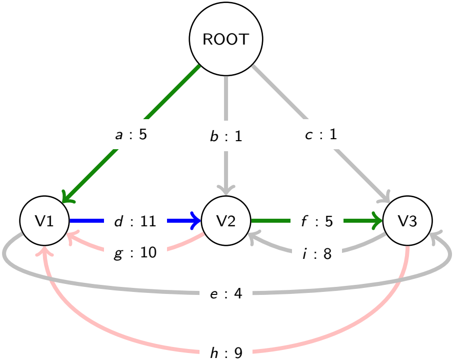
An Example - Expanding Stage



	bestInEdge
V1	a ~~g~~
V2	d
V3	f
V4	a ~~h~~
V5	a

	kicksOut
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

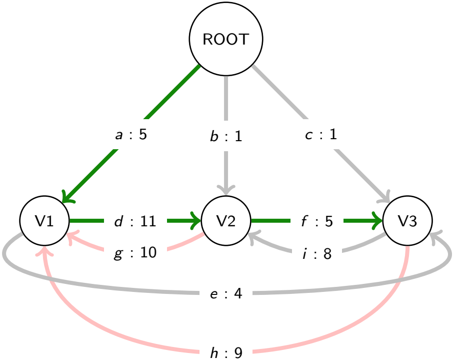
An Example - Expanding Stage



	`bestInEdge`
V1	a ~~g~~
V2	d
V3	f
V4	a ~~h~~
V5	a

	`kicksOut`
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

An Example - Expanding Stage



	`bestInEdge`
V1	a ~~g~~
V2	d
V3	f
V4	a ~~h~~
V5	a

	`kicksOut`
a	g, h
b	d, h
c	f
d	
e	f
f	
g	
h	g
i	d

Summing over all non-projective trees

~~Finding highest-scoring non-projective tree~~

- Consider the sentence "John saw Mary (left)".
- The Chu-Liu-Edmonds algorithm finds the maximum-weight spanning tree – may be non-projective.
- Can be found in time $O(n^2)$.

- How about total weight Z of all trees?
- Can be found in time $O(n^3)$ by matrix determinants and inverses (Smith & Smith, 2007).

Graph Theory to the Rescue!

$O(n^3)$ time!

Tutte's **Matrix-Tree Theorem** (1948)

The **determinant** of the Kirchoff (aka Laplacian) adjacency matrix of directed graph *G* without row and column *r* is equal to the **sum of scores of all directed spanning trees** of *G* rooted at node *r*.

Exactly the *Z* we need!

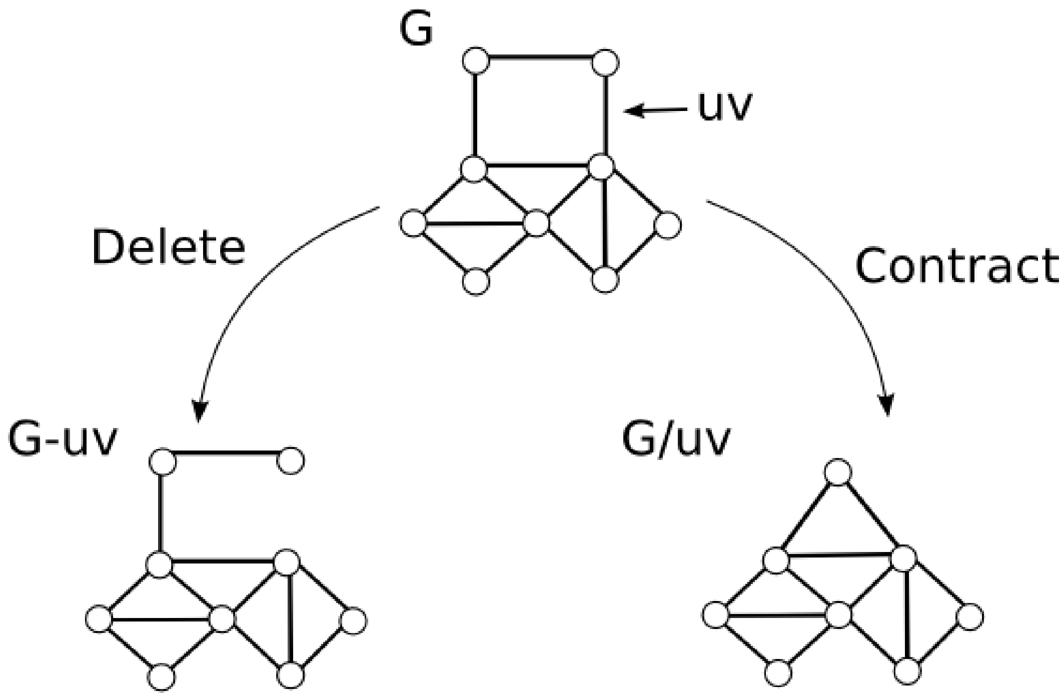
Building the Kirchoff (Laplacian) Matrix

$$\begin{vmatrix} \sum_{j\neq 1} s(1,j) & -s(2,1) & \cdots & -s(n,1) \\ -s(1,2) & \sum_{j\neq 2} s(2,j) & \cdots & -s(n,2) \\ \vdots & \vdots & \ddots & \vdots \\ -s(1,n) & -s(2,n) & \cdots & \sum_{j\neq n} s(n,j) \end{vmatrix}$$

- Negate edge scores
- Sum columns (children)
- Strike root row/col.
- Take determinant

N.B.: This allows multiple children of root, but see Koo et al. 2007.

Graph Deletion & Contraction



Important fact: κ(G) = κ(G-{e}) + κ(G\{e})

Why Should This Work?

Clear for 1x1 matrix; use induction

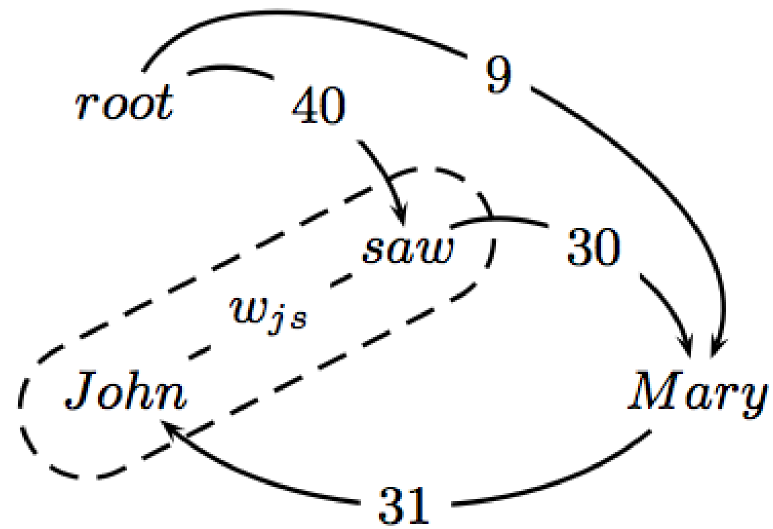
$$\begin{vmatrix} \sum_{j\neq 1} s(1,j) & -s(2,1) & \cdots & -s(n,1) \\ -s(1,2) & \sum_{j\neq 2} s(2,j) & \cdots & -s(n,2) \\ \vdots & \vdots & \ddots & \vdots \\ -s(1,n) & -s(2,n) & \cdots & \sum_{j\neq n} s(n,j) \end{vmatrix}$$

$K' \equiv K$ with contracted edge 1,2

$K'' \equiv K$ with deleted edge 1,2

$$|K| = s(1,2)|K'| + |K''|$$

Chu-Liu-Edmonds analogy:
Every node selects best parent
If cycles, contract and recurse



Undirected case; special root cases for directed