Log-Linear Models with Structured Outputs (continued)

Introduction to Natural Language Processing
Computer Science 585—Fall 2009
University of Massachusetts Amherst

David Smith

Overview

- What computations do we need?
- Smoothing log-linear models
- MEMMs vs. CRFs again
  - Action-based parsing and dependency parsing

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data

$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$

2. Initialize all parameters to zero
3. Classify training data with current parameters; calculate expectations $E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$
4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$
5. Take a step in the direction of the gradient
6. Repeat from 3 until convergence

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data
$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$
2. Initialize all parameters to zero
3. Classify training data with current parameters; calculate expectations $$E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$$
4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$
5. Take a step in the direction of the gradient
6. Repeat from 3 until convergen

Where have we seen expected counts before?

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data
$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$
2. Initialize all parameters to zero
3. Classify training data with current parameters; calculate expectations $E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$
4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$
5. Take a step in the direction of the gradient
6. Repeat from 3 until convergen

EM!

Where have we seen expected counts before?

Gradient-Based Training

- λ <- λ + rate * Gradient(F)
- After all training examples? (batch)
- After every example? (on-line)
- Use second derivative?
- A big field: numerical optimization

Overfitting

- If we have too many features, we can choose weights to model the training data perfectly
- If we have a feature that only appears in spam training, not ham training, it will get weight ∞ to maximize p(spam | feature) at 1.
- These behaviors
  - Overfit the training data
  - Will probably do poorly on test data

Solutions to Overfitting

- Throw out rare features.
  - Require every feature to occur > 4 times, and > 0 times with ling, and > 0 times with spam.
- Only keep, e.g., 1000 features.
  - Add one at a time, always greedily picking the one that most improves performance on held-out data.
- Smooth the observed feature counts.
- Smooth the weights by using a prior.
  - max $p(\lambda|\text{data})$ = max $p(\lambda, \text{data}) = p(\lambda)p(\text{data}|\lambda)$
  - decree $p(\lambda)$ to be high when most weights close to 0

Smoothing with Priors

- What if we had a prior expectation that parameter values wouldn't be very large?
- We could then balance evidence suggesting large (or infinite) parameters against our prior expectation.
- The evidence would never totally defeat the prior, and parameters would be smoothed (and kept finite)
- We can do this explicitly by changing the optimization objective to maximum posterior likelihood:

  $\log P(y, \lambda \mid x) = \log P(\lambda) + \log P(y \mid x, \lambda)$

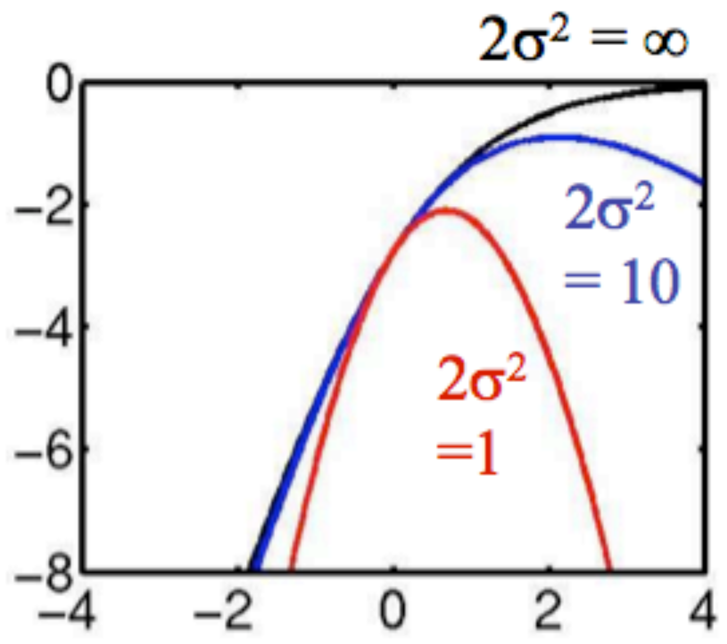
  Posterior Prior Likelihood

Smoothing: Priors

- Gaussian, or quadratic, priors:
  - Intuition: parameters shouldn't be large.
  - Formalization: prior expectation that each parameter will be distributed according to a gaussian with mean μ and variance σ^2.

$$P(\lambda_i) = \frac{1}{\sigma_i\sqrt{2\pi}}\exp\left(-\frac{(\lambda_i-\mu_i)^2}{2\sigma_i^2}\right)$$

  - Penalizes parameters for drifting to far from their mean prior value (usually μ=0).
  - $2\sigma^2$=1 works surprisingly well.



Parsing as Structured Prediction

Shift-reduce parsing

Stack	Input remaining	Action
()	Book that flight	shift
(Book)	that flight	reduce, Verb → book, (Choice #1 of 2)
(Verb)	that flight	shift
(Verb that)	flight	reduce, Det → that
(Verb Det)	flight	shift
(Verb Det flight)		reduce, Noun → flight
(Verb Det Noun)		reduce, NOM → Noun
(Verb Det NOM)		reduce, NP → Det NOM
(Verb NP)		reduce, VP → Verb NP
(Verb)		reduce, S → V
(S)		SUCCESS!

Ambiguity may lead to the need for backtracking.

Shift-reduce parsing

Stack	Input remaining	Action
()	Book that flight	shift
(Book)	that flight	reduce, Verb $\longrightarrow$ book, (Choice #1 of 2)
(Verb)	that flight	shift
(Verb that)	flight	reduce, Det $\longrightarrow$ that
(Verb Det)	flight	shift
(Verb Det flight)		reduce, Noun $\longrightarrow$ flight
(Verb Det Noun)		reduce, NOM $\longrightarrow$ Noun
(Verb Det NOM)		reduce, NP $\longrightarrow$ Det NOM
(Verb NP)		reduce, VP $\longrightarrow$ Verb NP
(Verb)		reduce, S $\longrightarrow$ V
(S)		SUCCESS!

~~Ambiguity may lead to the need for backtracking.~~

Shift-reduce parsing

Stack	Input remaining	Action
()	Book that flight	shift
(Book)	that flight	reduce, Verb $\rightarrow$ book, (Choice #1 of 2)
(Verb)	that flight	shift
(Verb that)	flight	reduce, Det $\rightarrow$ that
(Verb Det)	flight	shift
(Verb Det flight)		reduce, Noun $\rightarrow$ flight
(Verb Det Noun)		reduce, NOM $\rightarrow$ Noun
(Verb Det NOM)		reduce, NP $\rightarrow$ Det NOM
(Verb NP)		reduce, VP $\rightarrow$ Verb NP
(Verb)		reduce, S $\rightarrow$ V
(S)		SUCCESS!

~~Ambiguity may lead to the need for backtracking.~~

Train log-linear model of p(action | context)

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

⬇ Part-of-speech tagging

POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

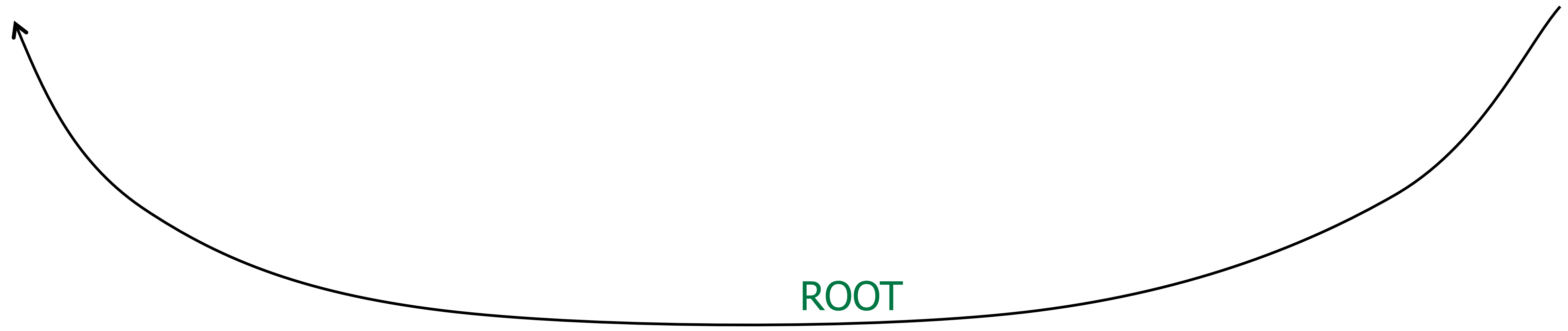
POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

Word dependency parsing

Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

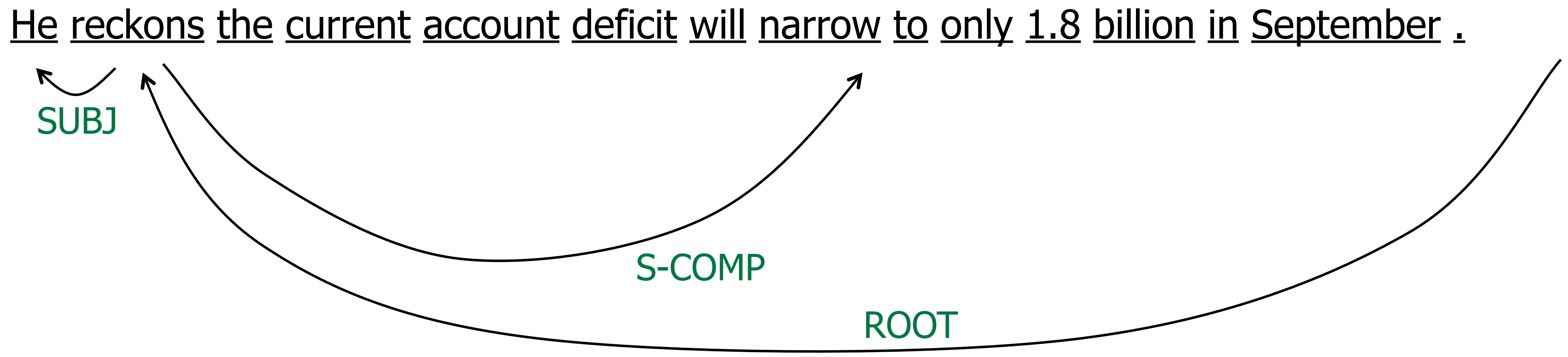
Part-of-speech tagging

POS-tagged sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.
PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

Word dependency parsing

Word dependency parsed sentence


He reckons the current account deficit will narrow to only 1.8 billion in September .

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

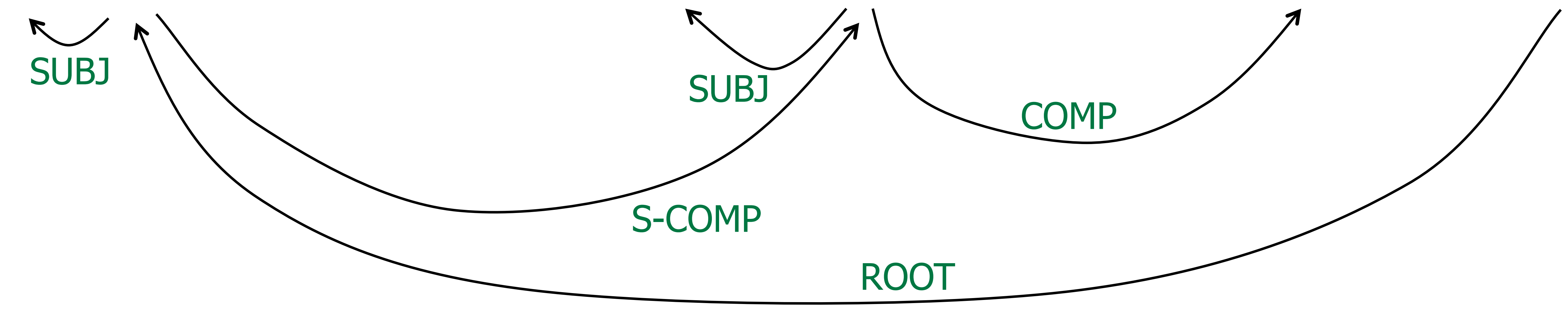
POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

Word dependency parsing

Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

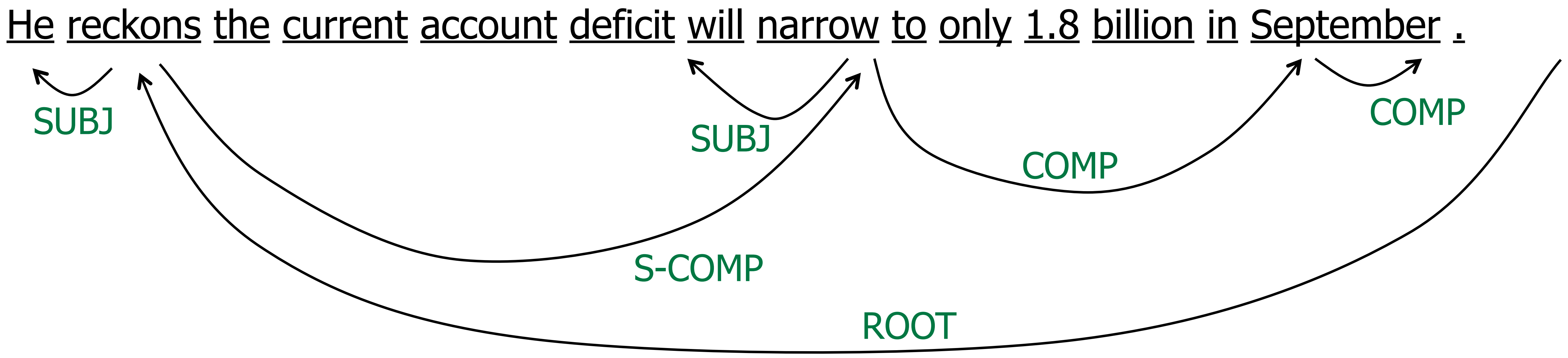
Part-of-speech tagging

POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

Word dependency parsing

Word dependency parsed sentence


He reckons the current account deficit will narrow to only 1.8 billion in September .
SUBJ
SUBJ
COMP
COMP
S-COMP
ROOT

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

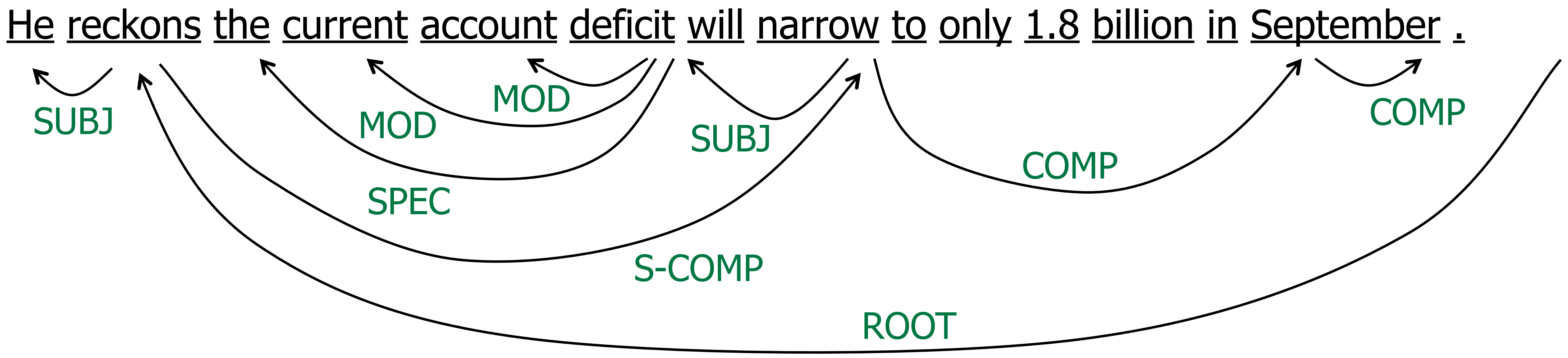
Part-of-speech tagging

POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

Word dependency parsing

Word dependency parsed sentence


He reckons the current account deficit will narrow to only 1.8 billion in September .
SUBJ
MOD
MOD
SUBJ
SPEC
COMP
COMP
S-COMP
ROOT

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

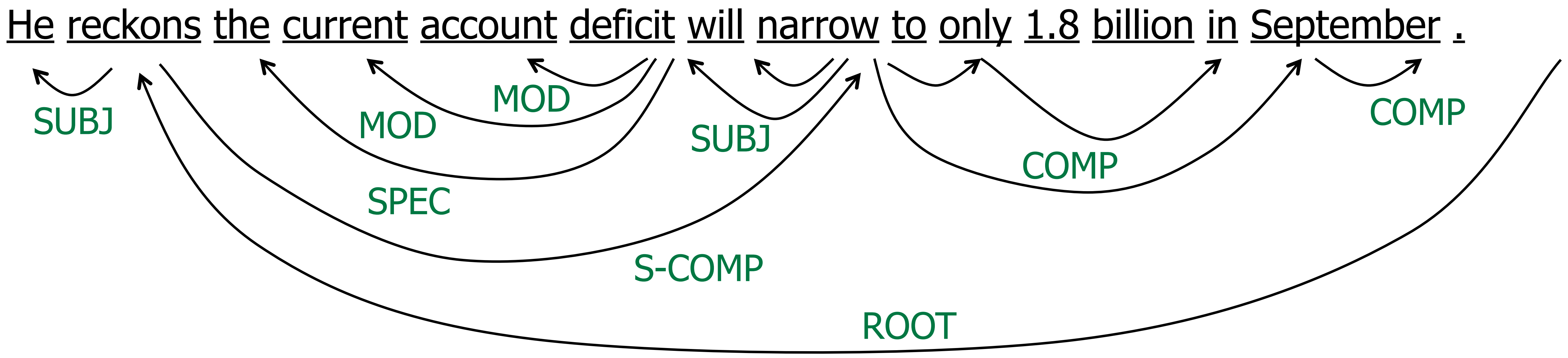
⬇ Part-of-speech tagging

POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

⬇ Word dependency parsing

Word dependency parsed sentence


He reckons the current account deficit will narrow to only 1.8 billion in September .
SUBJ
MOD
MOD
SUBJ
SPEC
S-COMP
COMP
COMP
ROOT

slide adapted from Yuji Matsumoto

Word Dependency Parsing

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

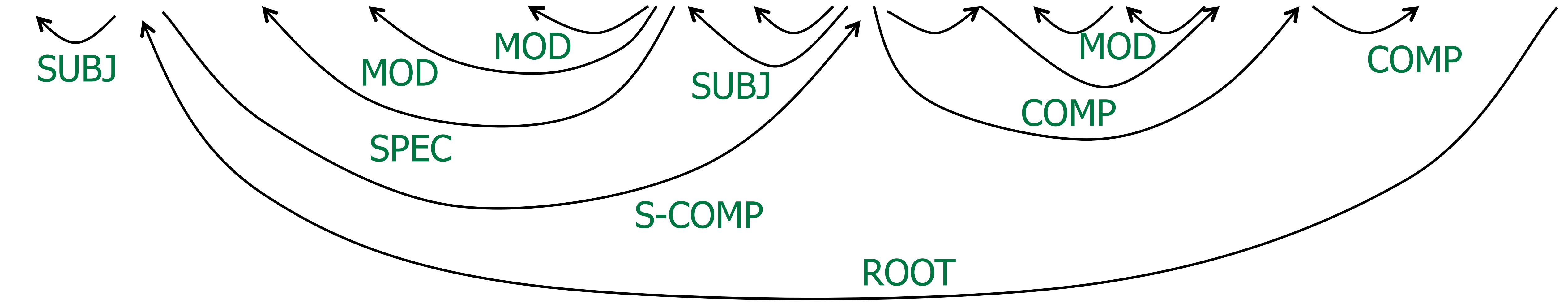
POS-tagged sentence

He	reckons	the	current	account	deficit	will	narrow	to	only	1.8	billion	in	September	.
PRP	VBZ	DT	JJ	NN	NN	MD	VB	TO	RB	CD	CD	IN	NNP	.

Word dependency parsing

Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



slide adapted from Yuji Matsumoto

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$$p(A) * p(B \mid A) * p(C \mid A,B) * p(D \mid A,B,C) * \ldots$$

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$$

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$$

each choice depends on a limited part of the history

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$

each choice depends on a limited part of the history

but which dependencies to allow? $p(D \mid A,\cancel{B},C)$?

what if they're all worthwhile? $p(D \mid A,B,\cancel{C})$?

$\ldots p(D \mid A,B) * p(C \mid \cancel{A},B,D)$?

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$

which dependencies to allow? (given limited training data)

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$$

which dependencies to allow? (given limited training data)

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$

which dependencies to allow? (given limited training data)

$(1/Z) * \Phi(A) * \Phi(B,A) * \Phi(C,A) * \Phi(C,B)$

$* \Phi(D,A,B) * \Phi(D,B,C) * \Phi(D,A,C) *$

throw them all in!

…

Great ideas in NLP: Log-linear models

(Berger, della Pietra, della Pietra 1996; Darroch & Ratcliff 1972)

- In the beginning, we used generative models.

$$p(A) * p(B \mid A) * p(C \mid \cancel{A},B) * p(D \mid \cancel{A},B,C) * \ldots$$

which dependencies to allow? (given limited training data)

- Solution: Log-linear (max-entropy) modeling

$$(1/Z) * \Phi(A) * \Phi(B,A) * \Phi(C,A) * \Phi(C,B)$$

$$* \boxed{\Phi(D,A,B) * \Phi(D,B,C) * \Phi(D,A,C)} * \ldots$$

throw them all in!

  - Features may interact in arbitrary ways
  - **Iterative scaling** keeps adjusting the feature weights until the model agrees with the training data.

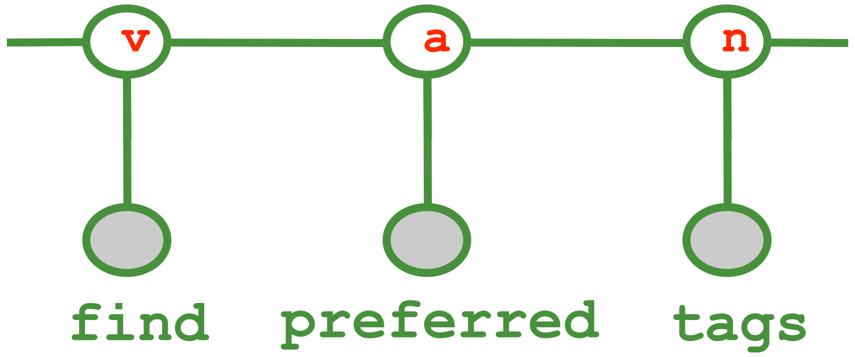
How about structured outputs?

How about structured outputs?

- Log-linear models great for n-way classification

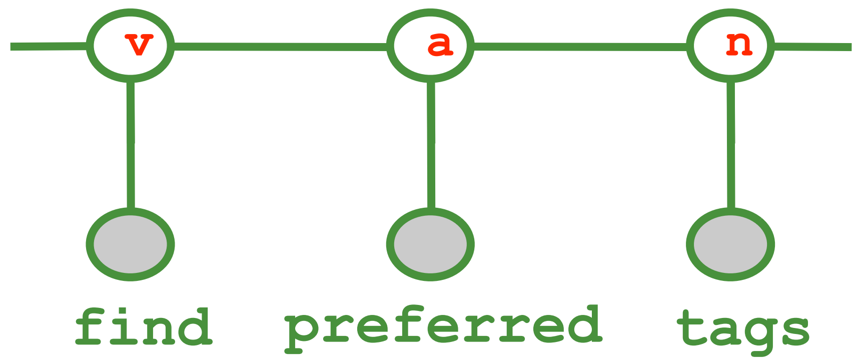
How about structured outputs?

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



How about structured outputs?

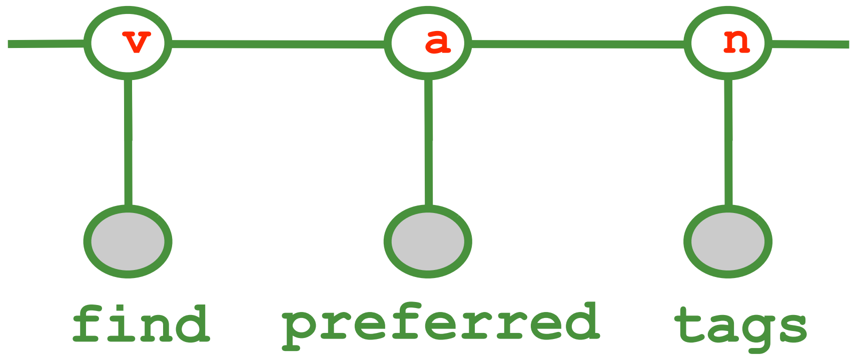
- Log-linear models great for n-way classification
- Also good for predicting sequences



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

How about structured outputs?

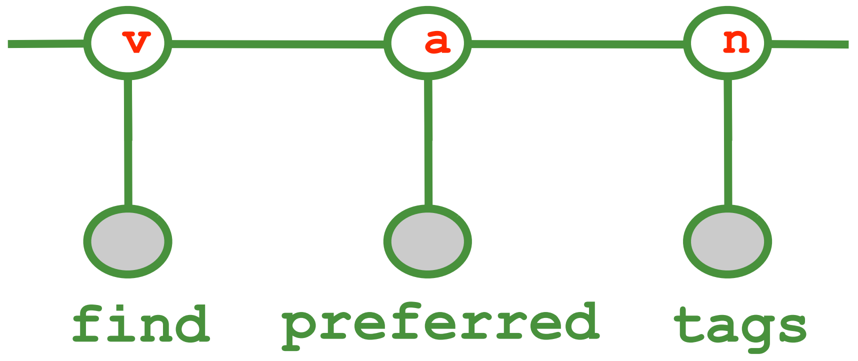
- Log-linear models great for n-way classification
- Also good for predicting sequences



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

How about structured outputs?

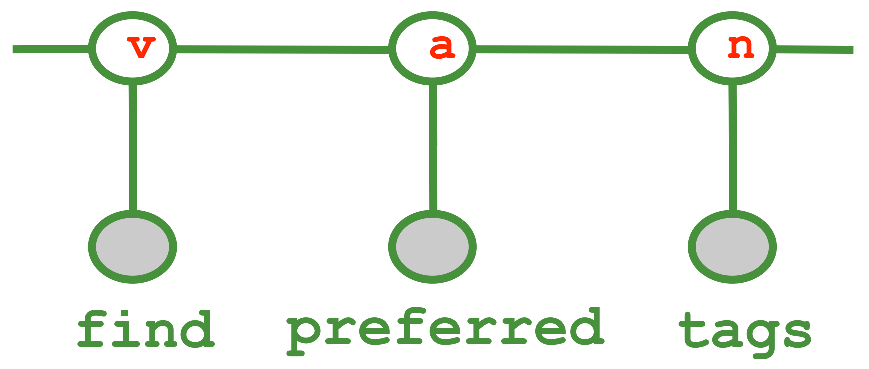
- Log-linear models great for n-way classification
- Also good for predicting sequences



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

How about structured outputs?

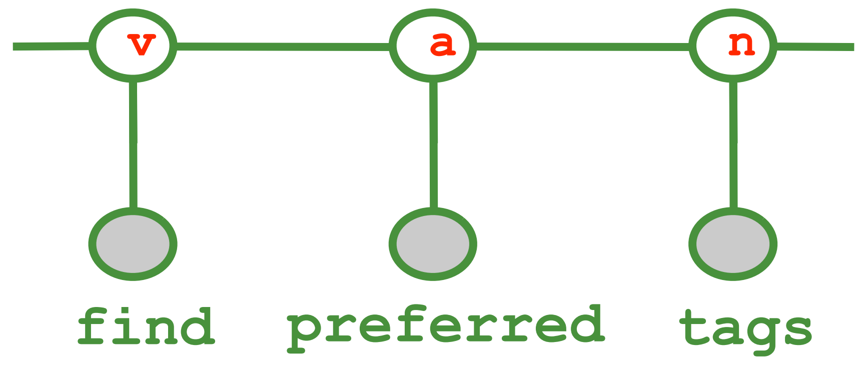
- Log-linear models great for n-way classification
- Also good for predicting sequences



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

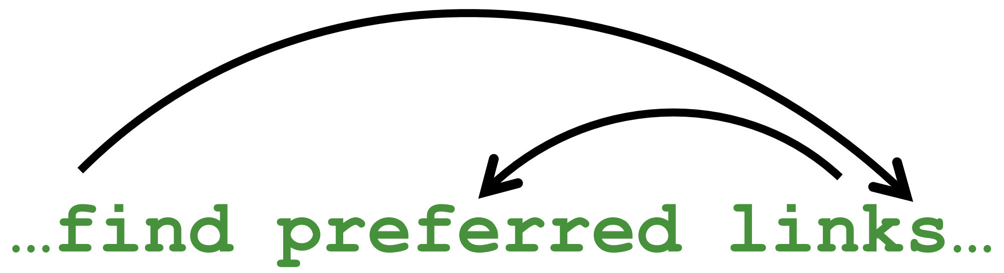
How about structured outputs?

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



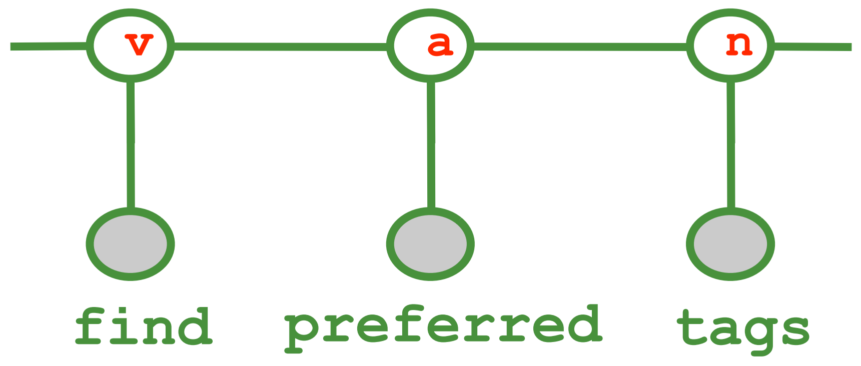
but to allow fast dynamic programming,
only use **n-gram** features

- Also good for dependency parsing



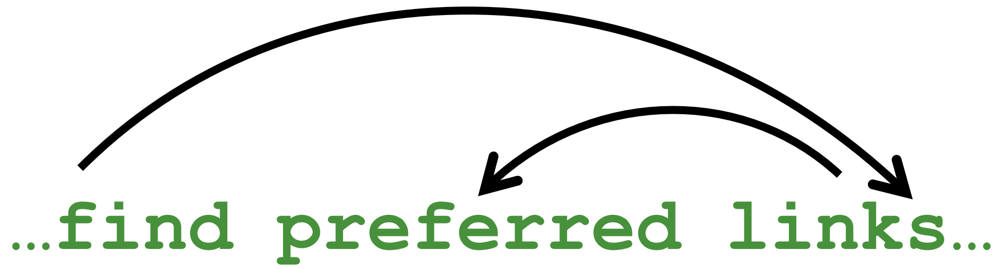
How about structured outputs?

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



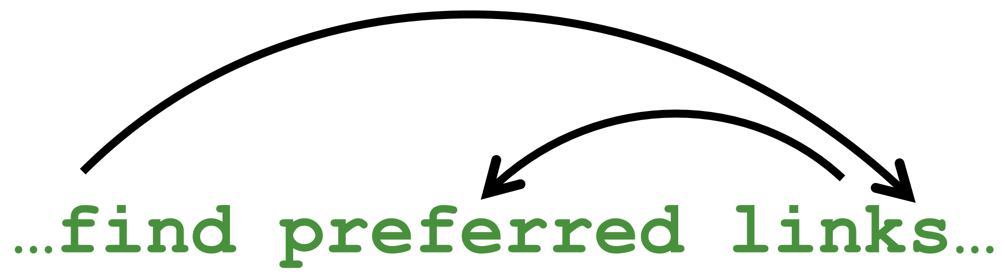
but to allow fast dynamic programming,
only use **n-gram** features

- Also good for dependency parsing



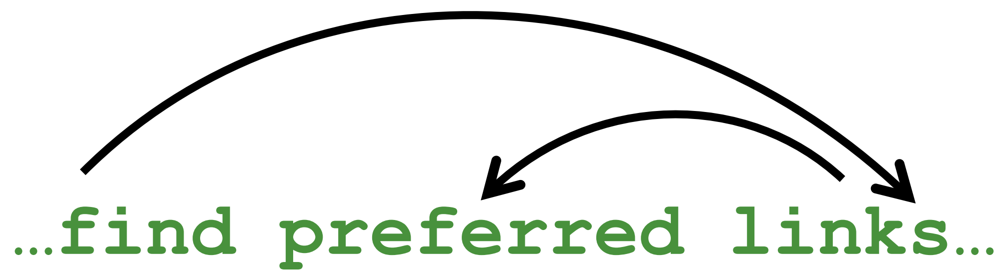
but to allow fast dynamic programming or MST parsing,
only use **single-edge** features

How about structured outputs?

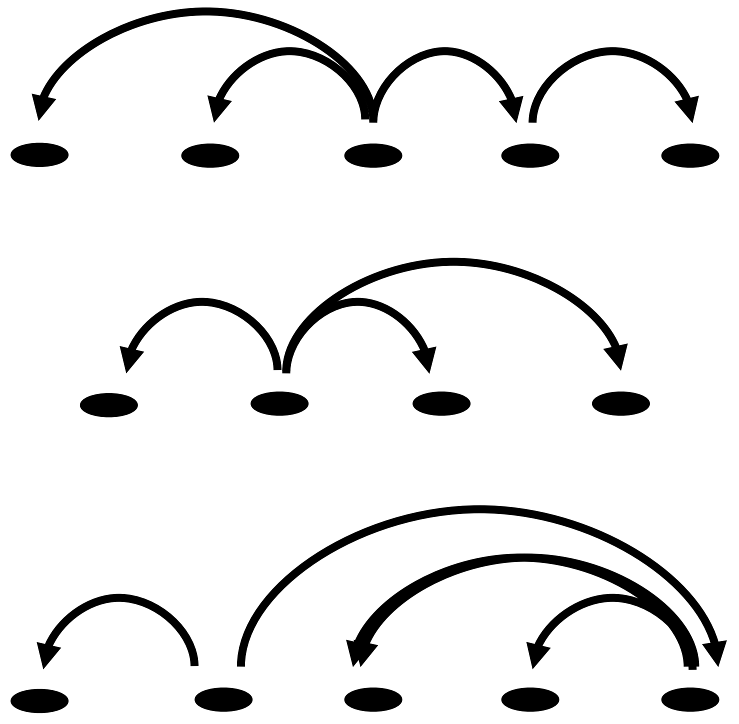


but to allow fast dynamic programming or MST parsing, only use **single-edge** features

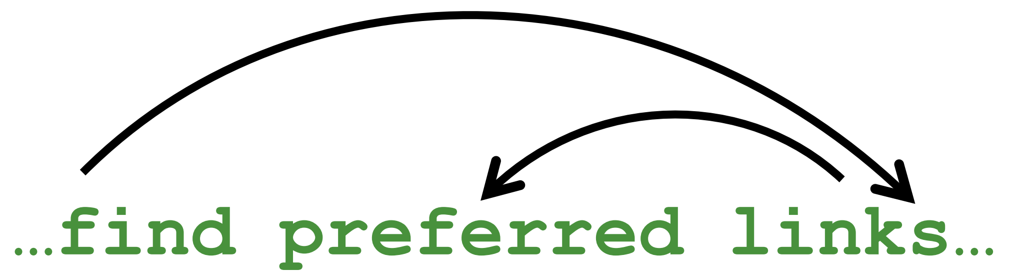
How about structured outputs?



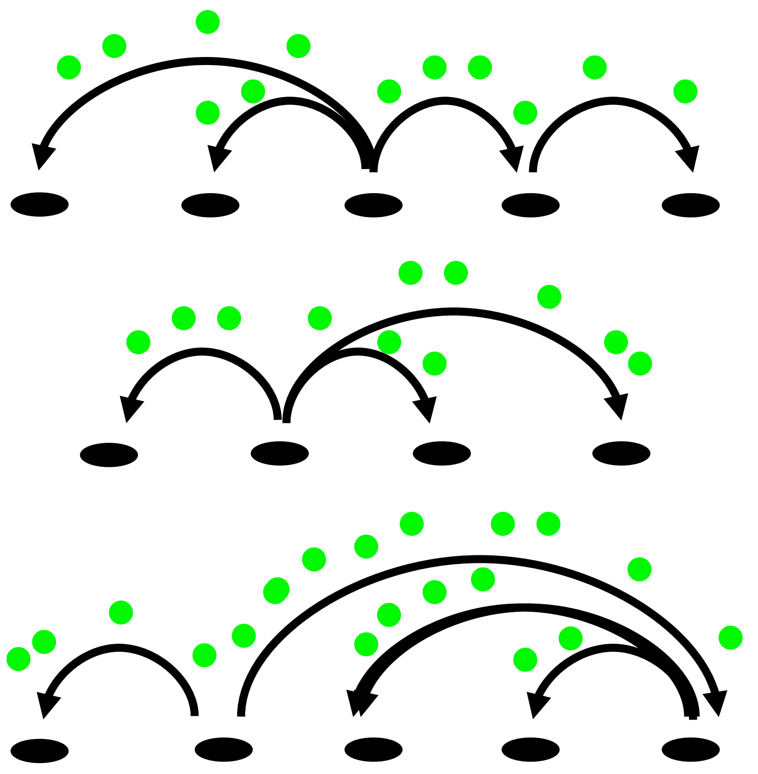
but to allow fast dynamic programming or MST parsing, only use **single-edge** features



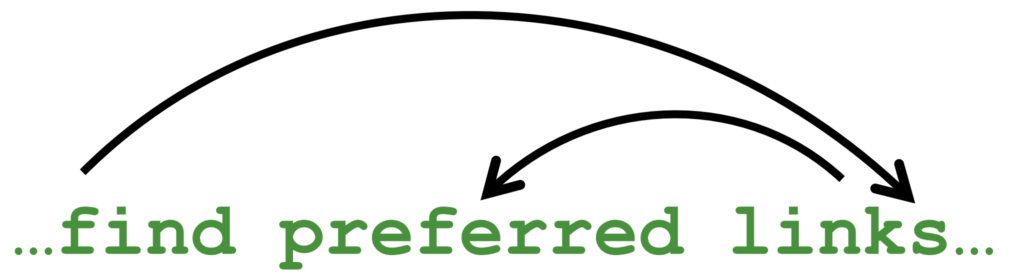
How about structured outputs?



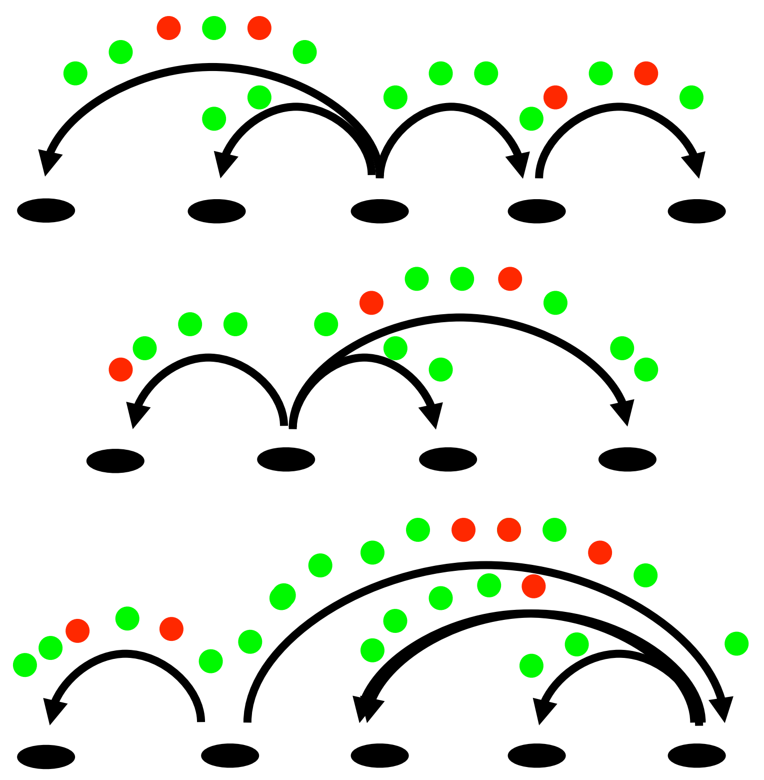
but to allow fast dynamic programming or MST parsing, only use **single-edge** features



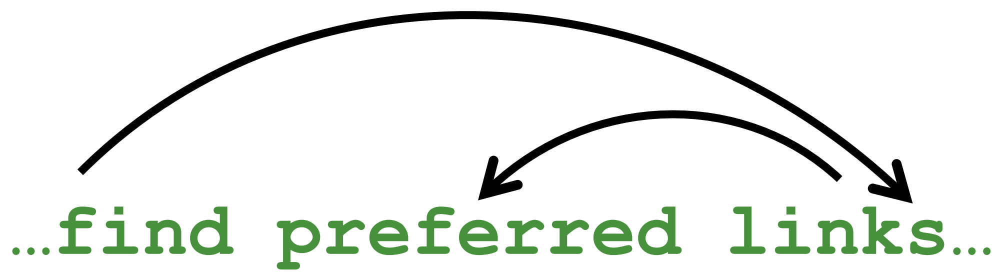
How about structured outputs?



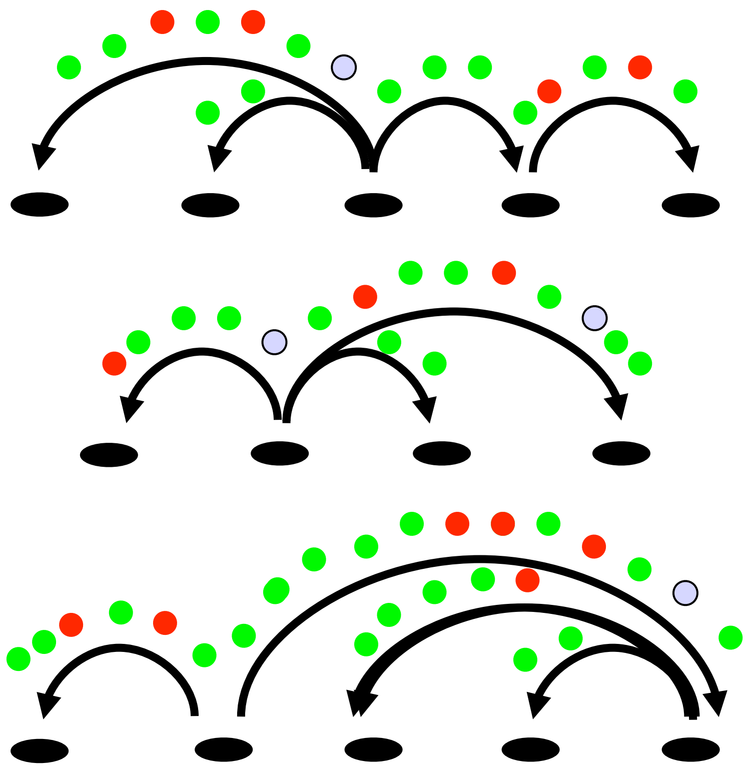
but to allow fast dynamic programming or MST parsing, only use **single-edge** features



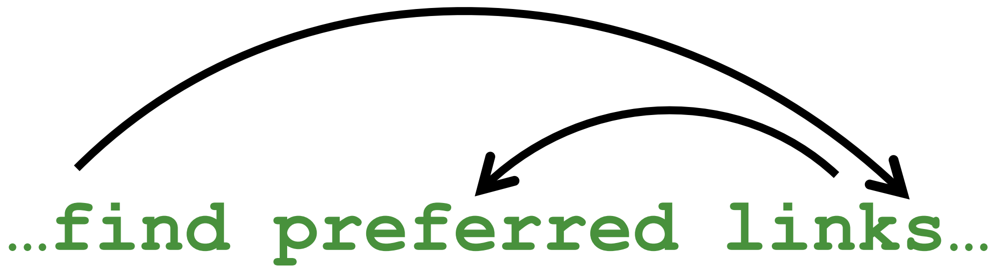
How about structured outputs?



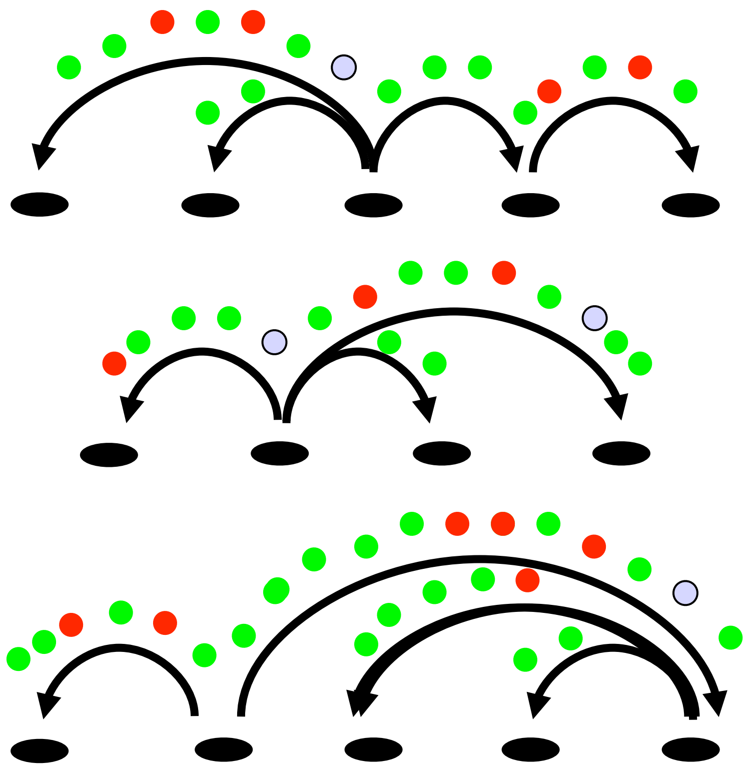
but to allow fast dynamic programming or MST parsing, only use **single-edge** features

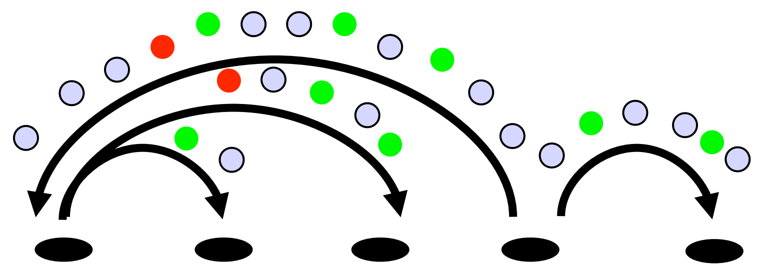


How about structured outputs?



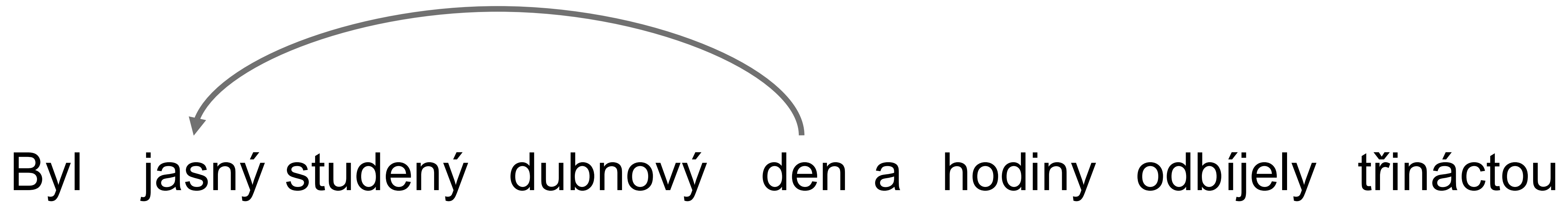
but to allow fast dynamic programming or MST parsing, only use **single-edge** features





Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

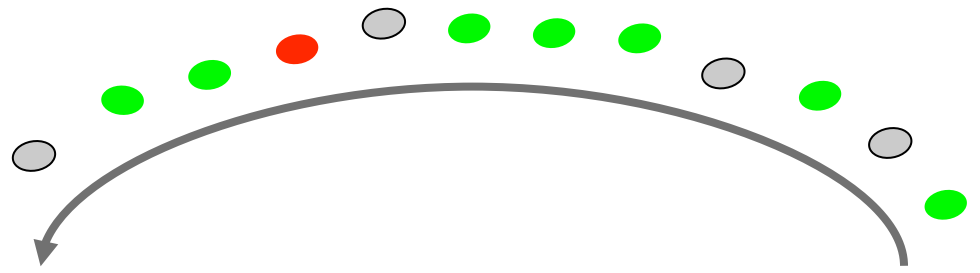


"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- Is this a good edge?

yes, lots of green ...

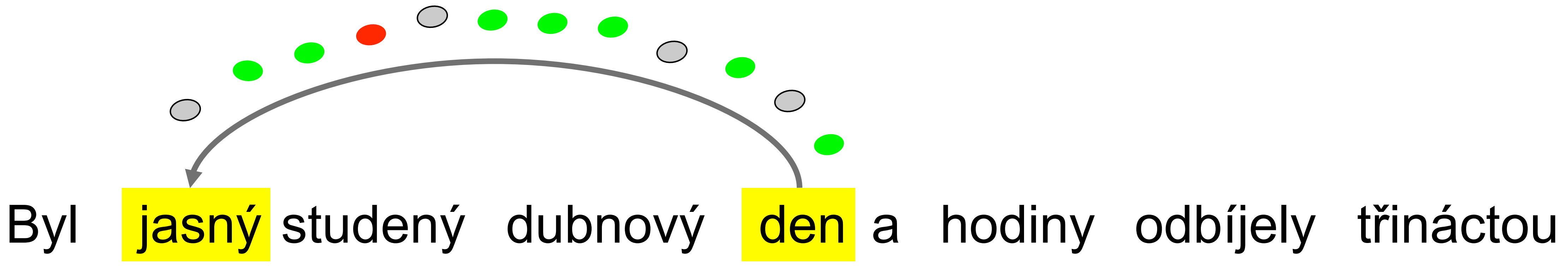


Byl jasný studený dubnový den a hodiny odbíjely třináctou

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

Edge-Factored Parsers (McDonald et al. 2005)

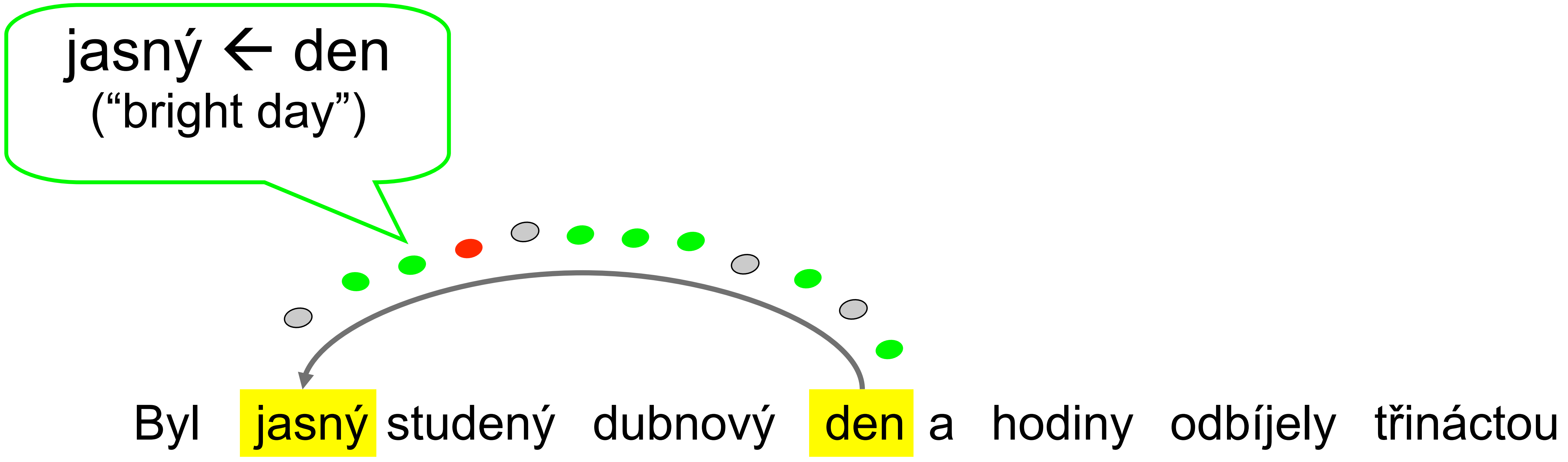
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

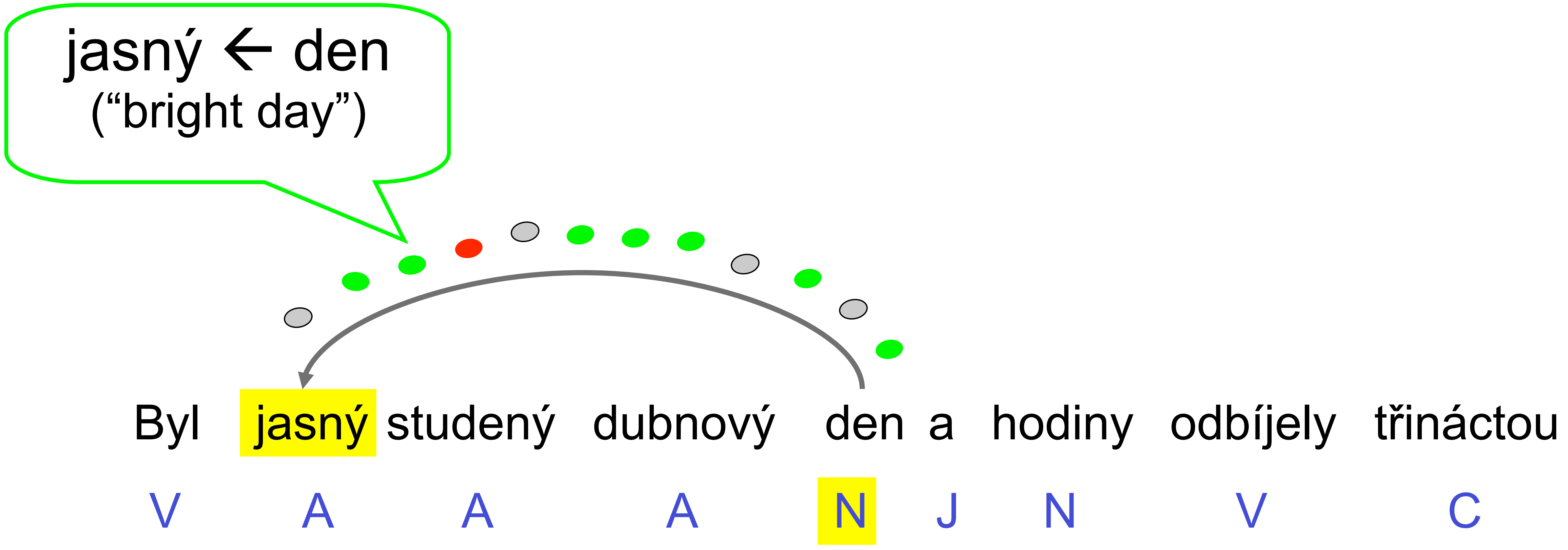
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

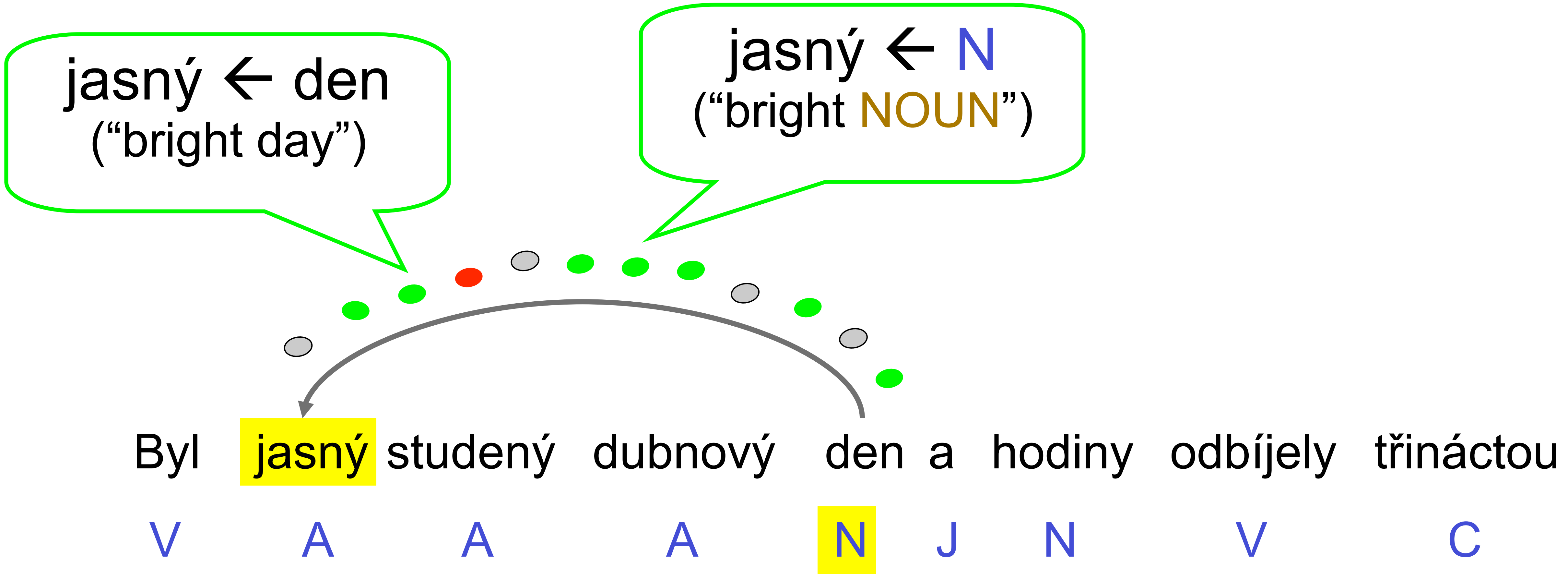
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

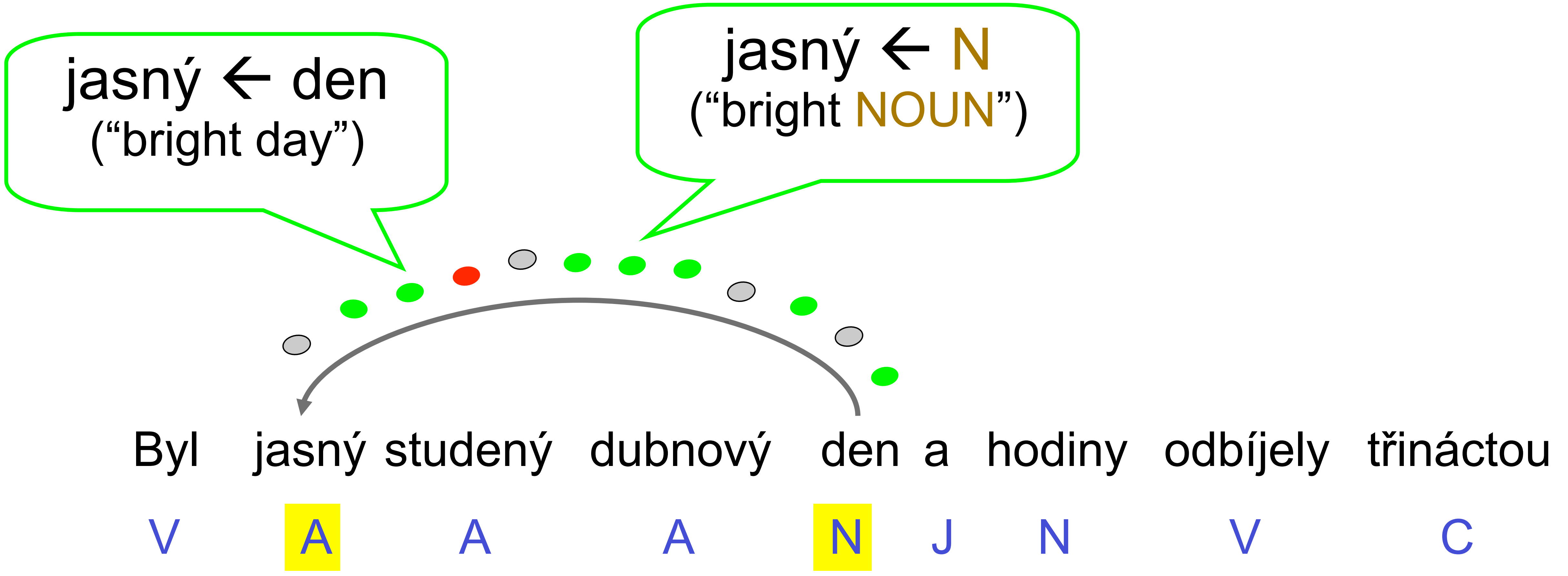
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

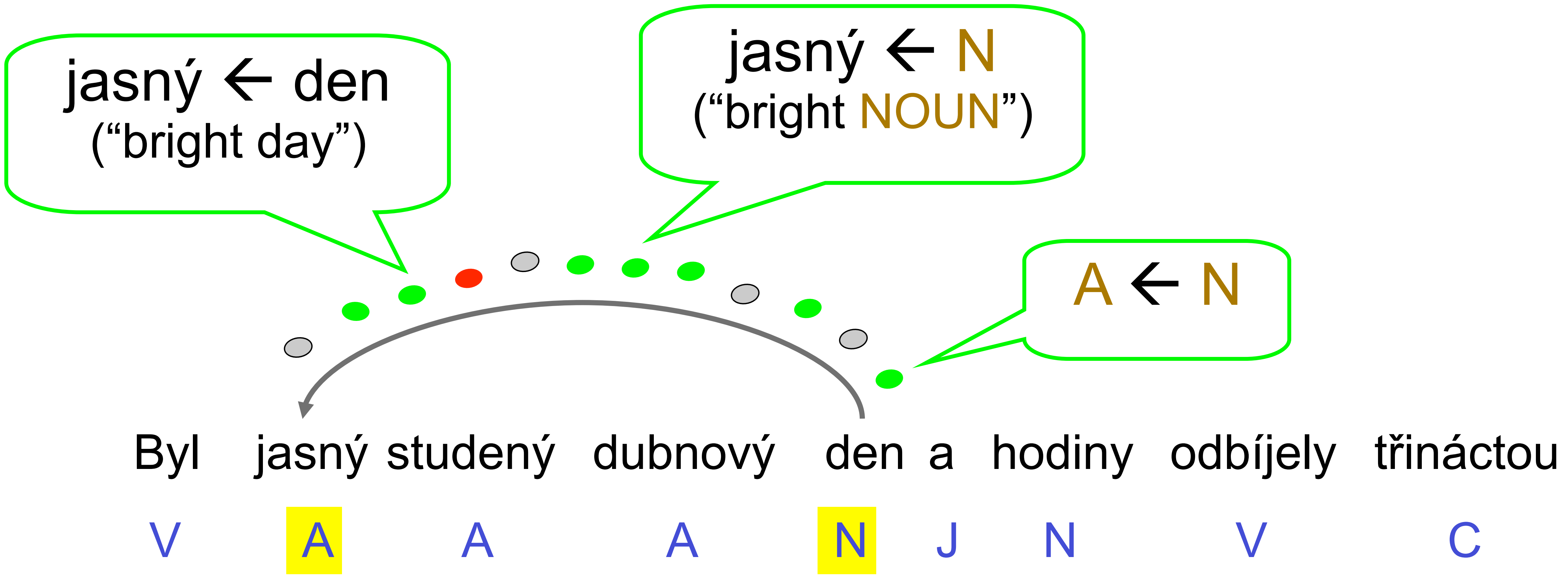
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

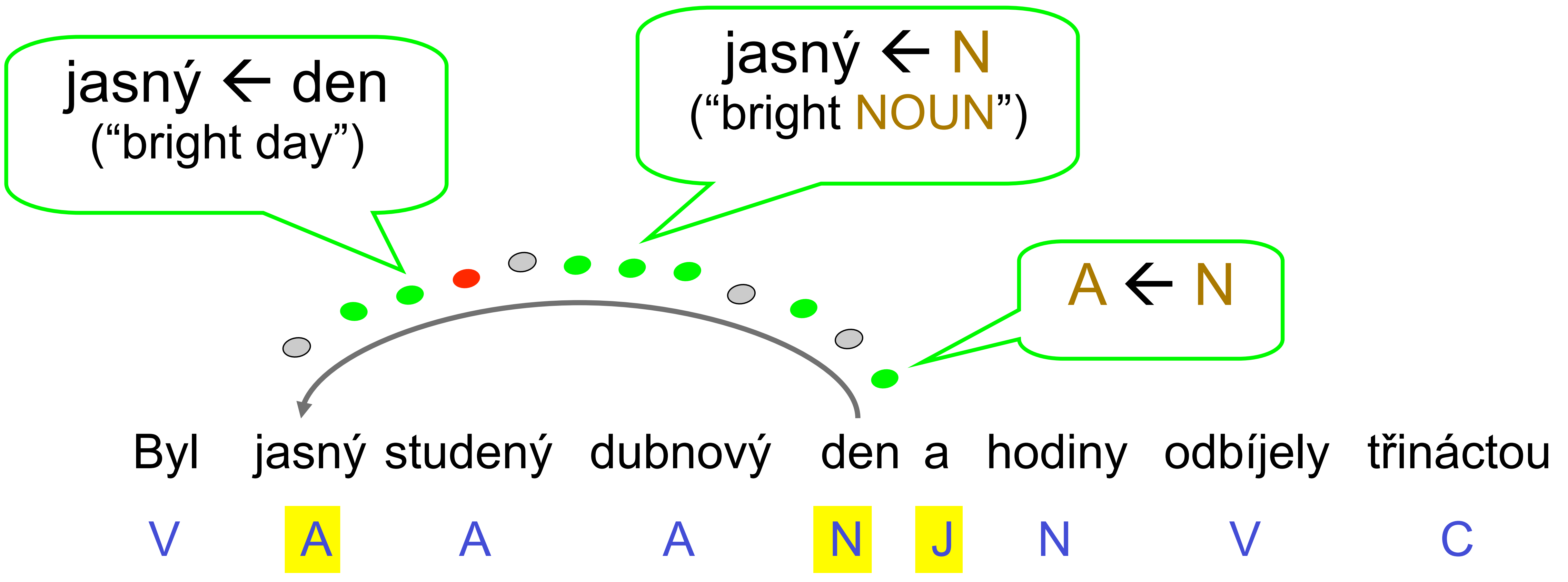
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

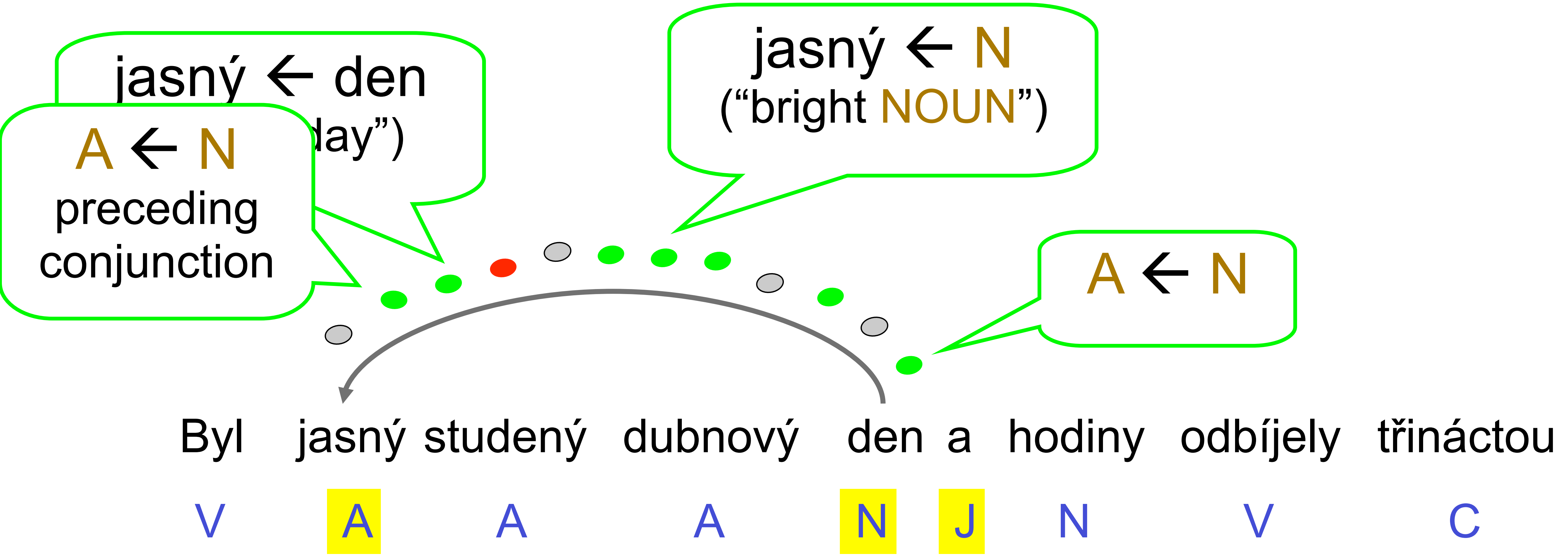
- Is this a good edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

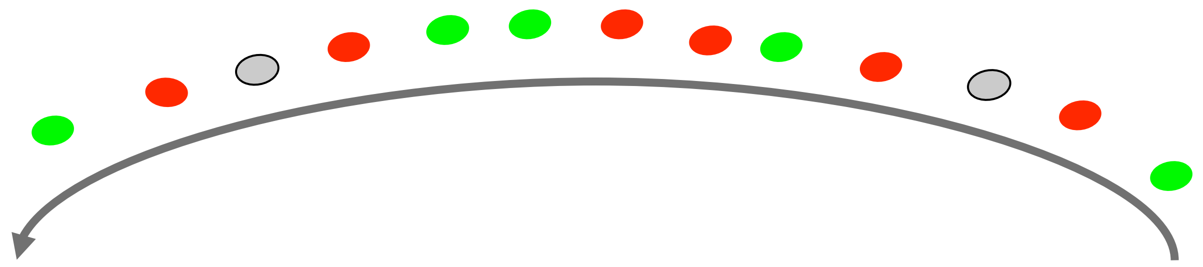
- Is this a good edge?



"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?

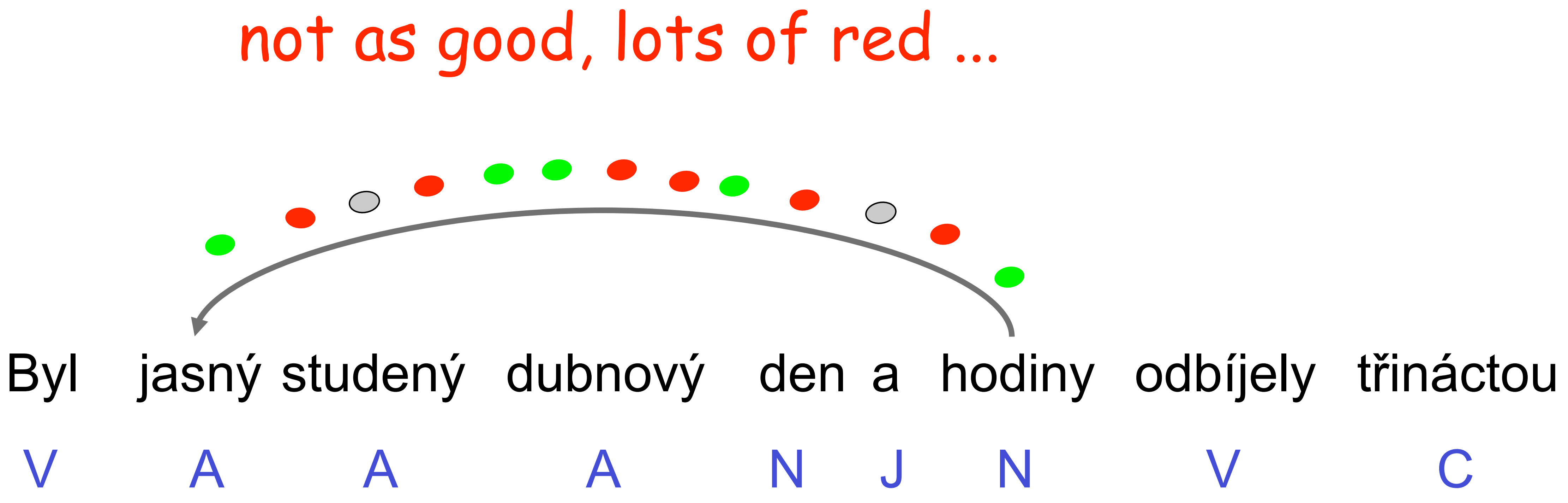


Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C

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

Edge-Factored Parsers (McDonald et al. 2005)

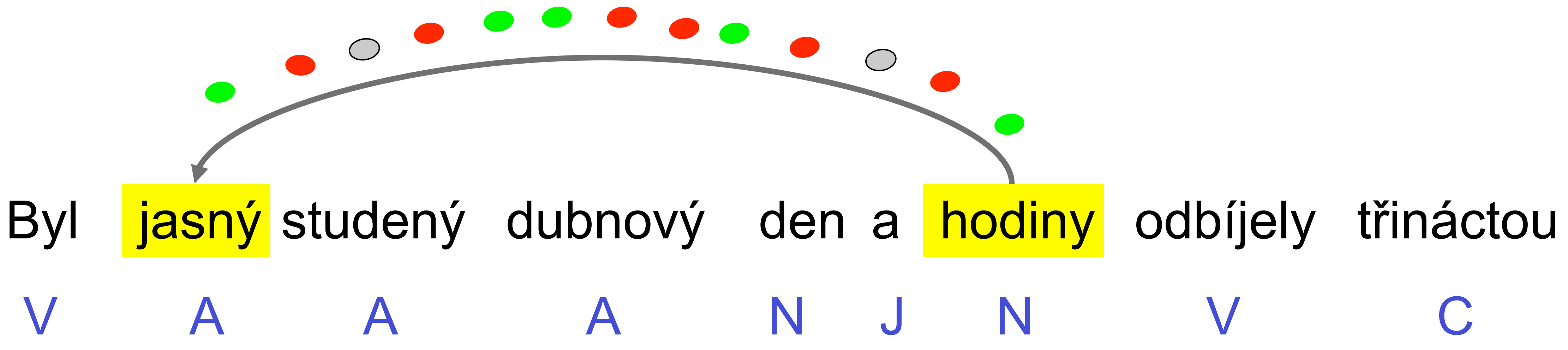
- How about this competing edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

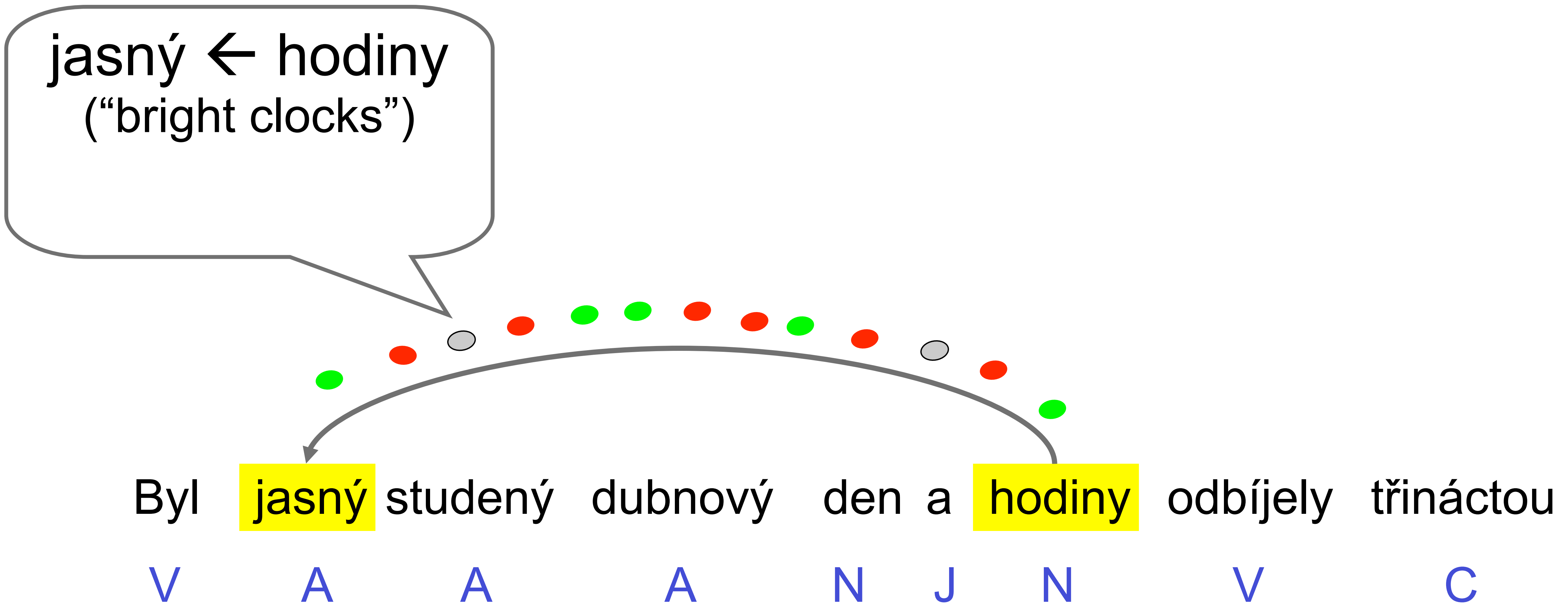
- How about this competing edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

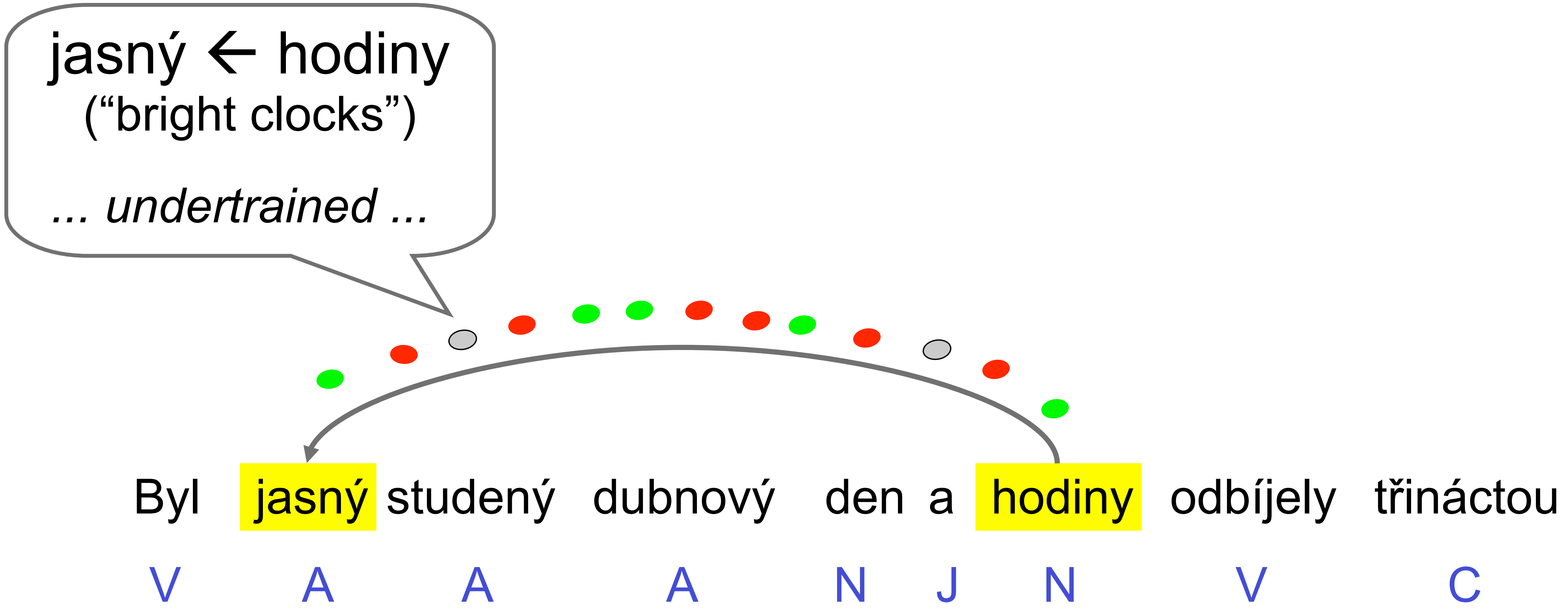
- How about this competing edge?



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

Edge-Factored Parsers (McDonald et al. 2005)

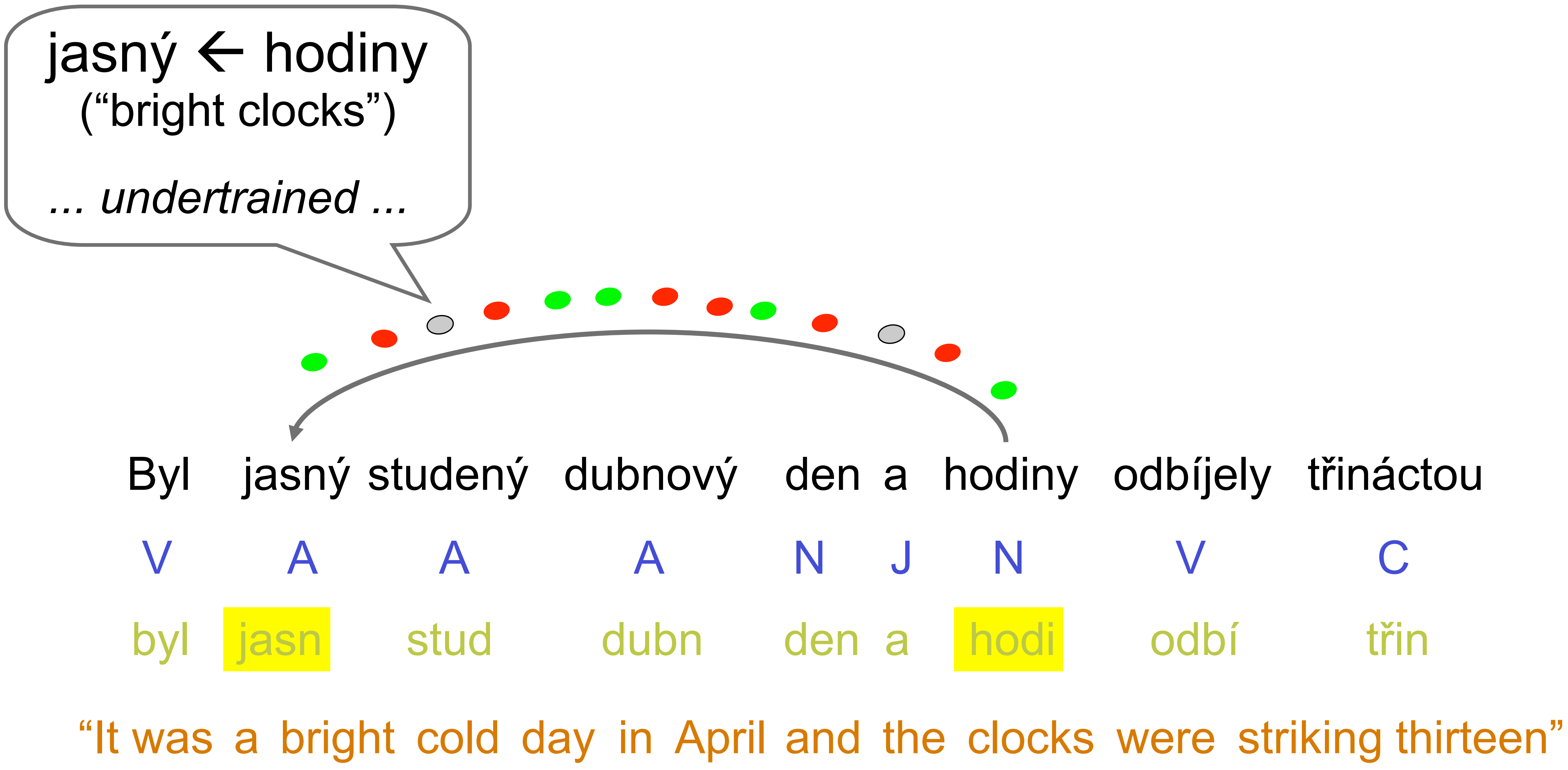
- How about this competing edge?



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

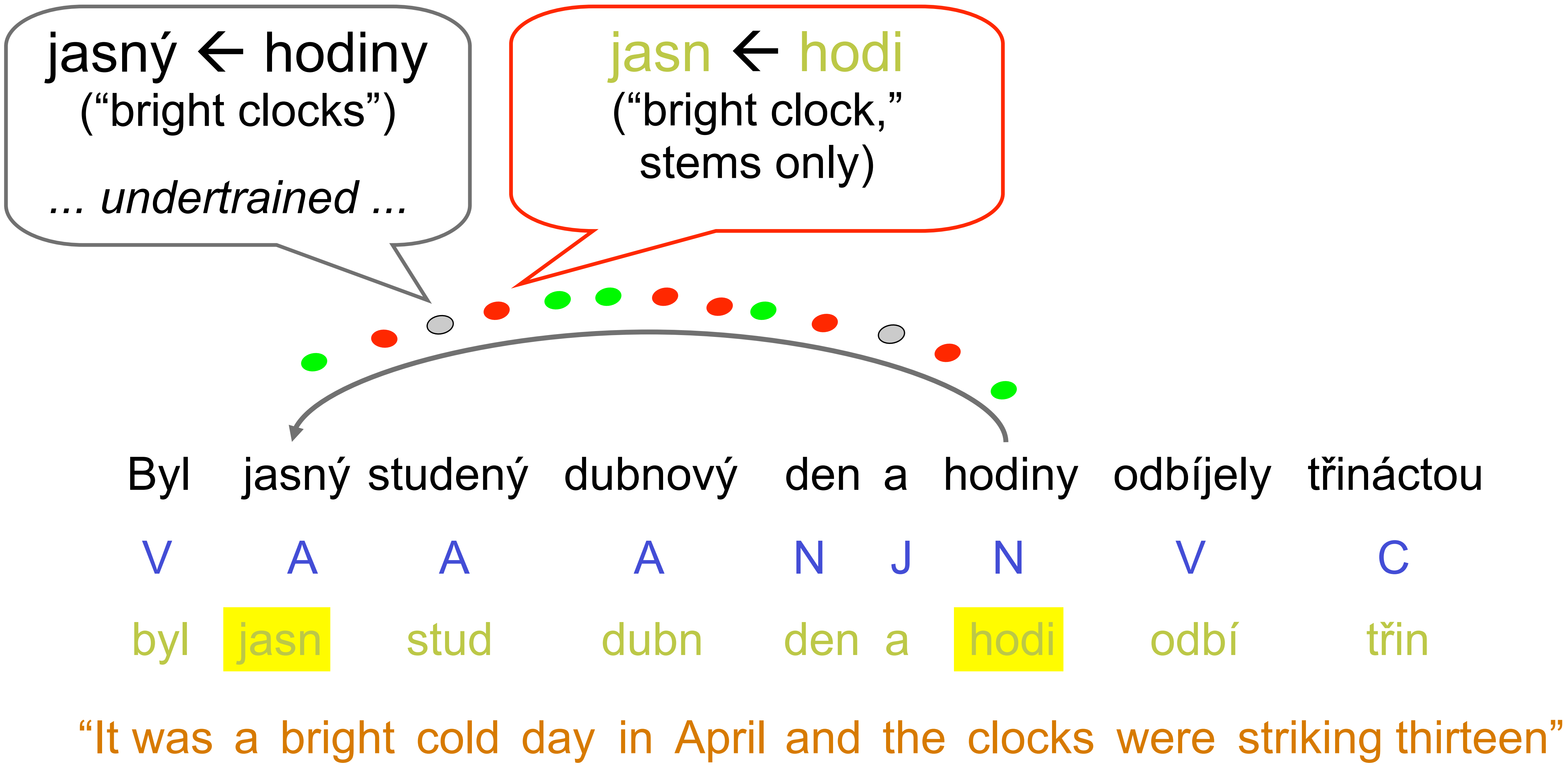
Edge-Factored Parsers (McDonald et al. 2005)

- 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
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)

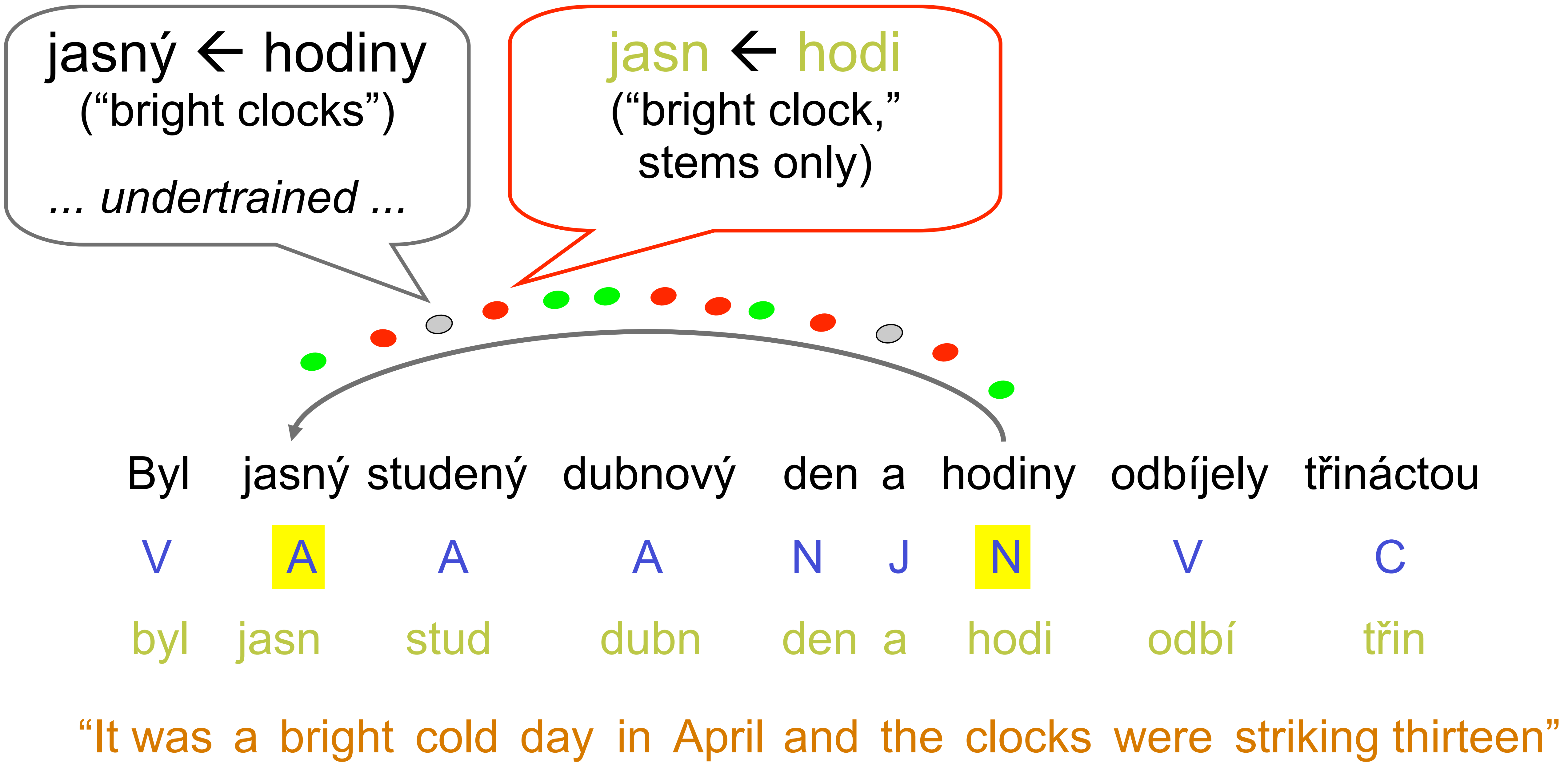
- How about this competing edge?



jasný ← hodiny
(“bright clocks”)
... undertrained ...
jasn ← hodi
(“bright clock,” stems only)
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”

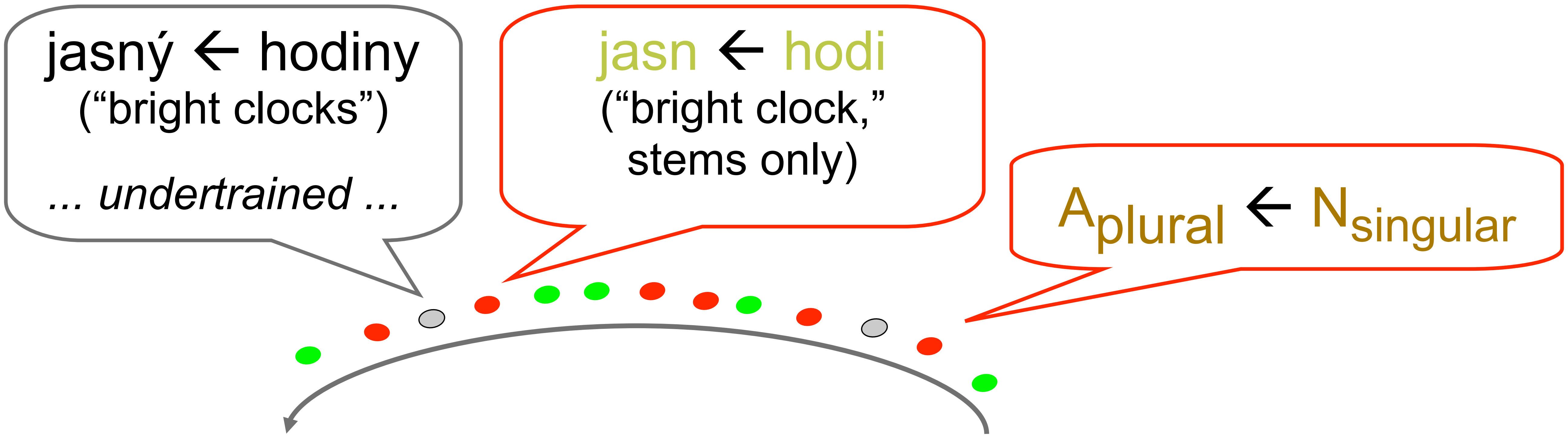
Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?


jasný ← hodiny
("bright clocks")
... *undertrained* ...
jasn ← hodi
("bright clock," stems only)
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?

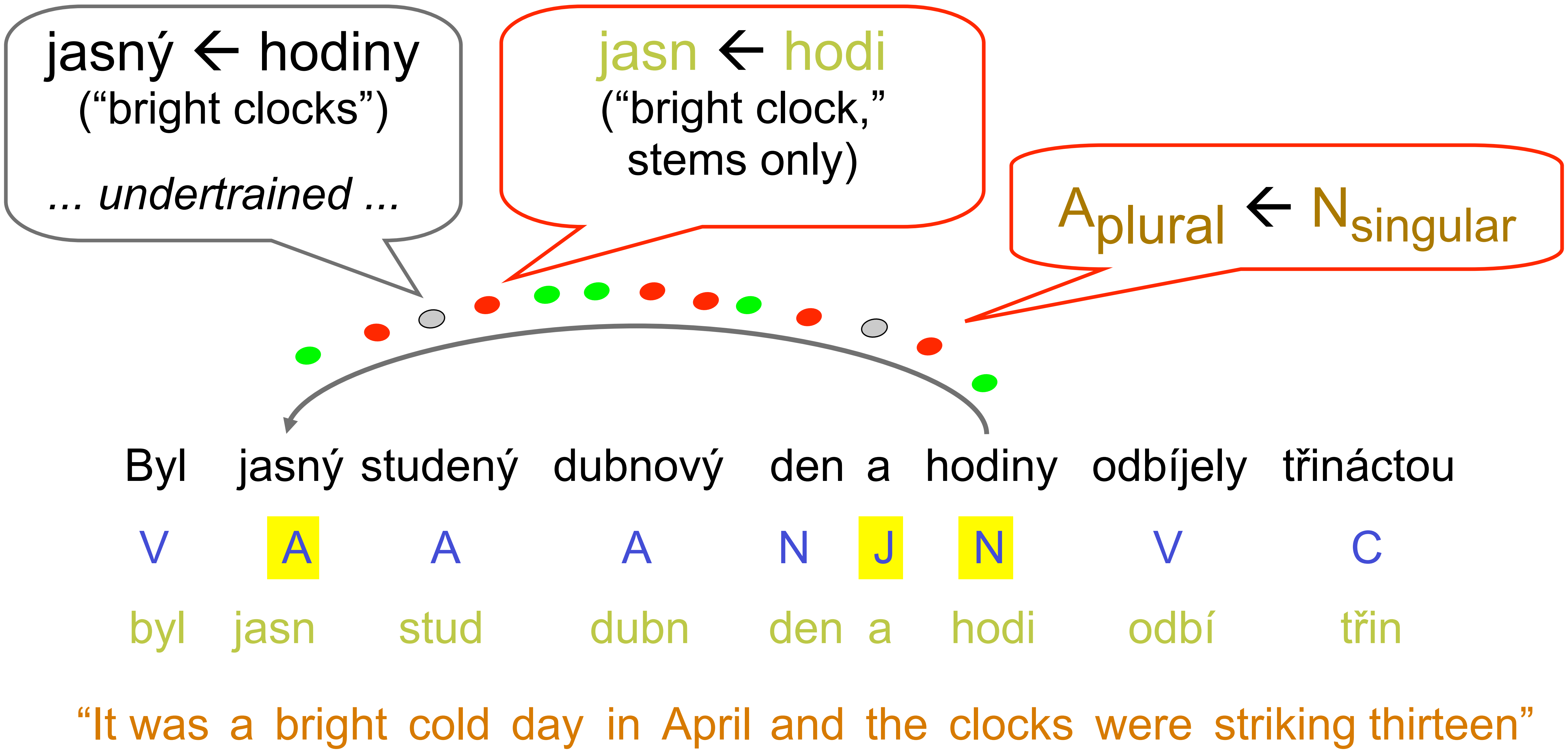

jasný ← hodiny
("bright clocks")
... *undertrained* ...
jasn ← hodi
("bright clock," stems only)
$A_{plural} \leftarrow N_{singular}$

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"

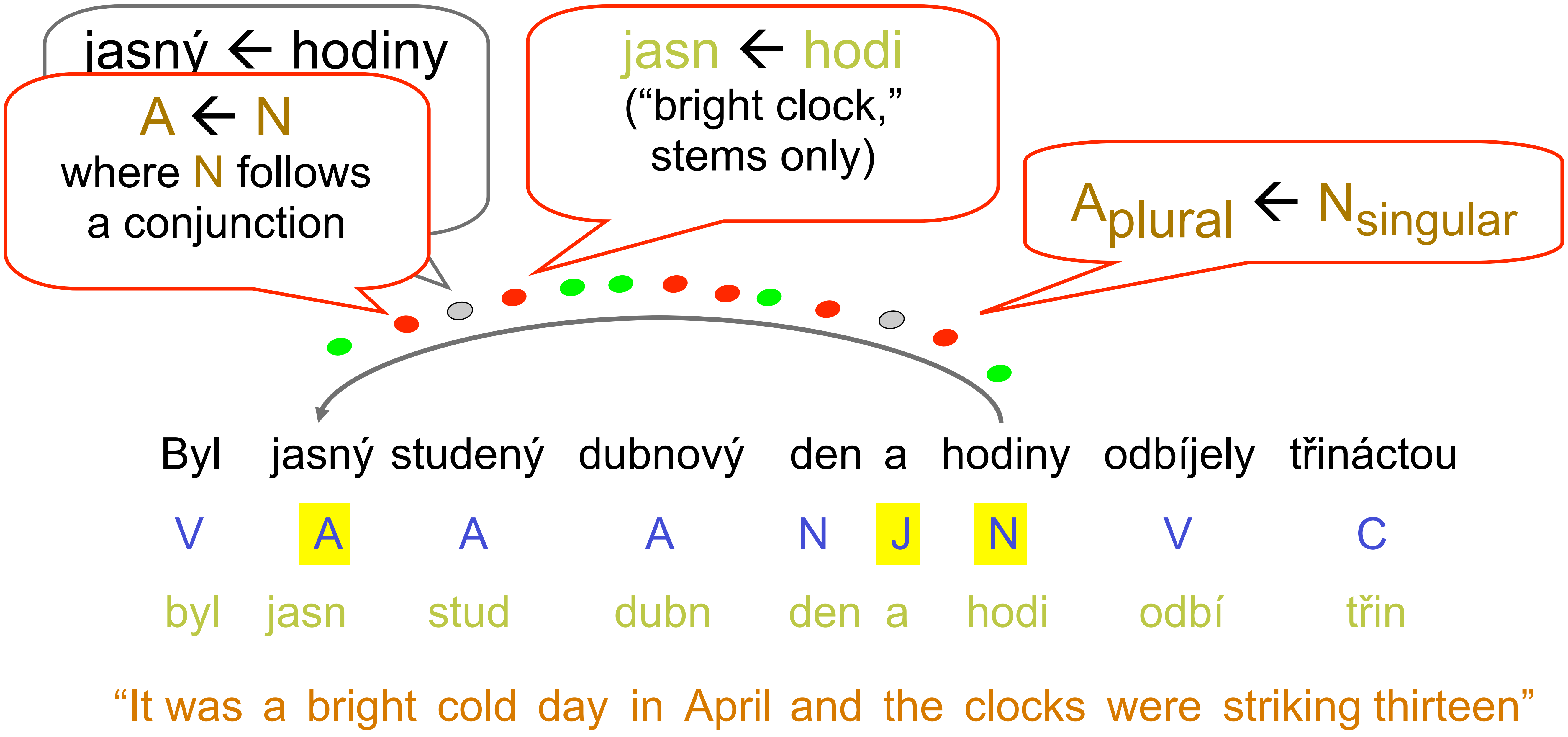
Edge-Factored Parsers (McDonald et al. 2005)

- How about this competing edge?


jasný ← hodiny
("bright clocks")
... *undertrained* ...
jasn ← hodi
("bright clock," stems only)
A_{plural} ← $N_{singular}$
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)

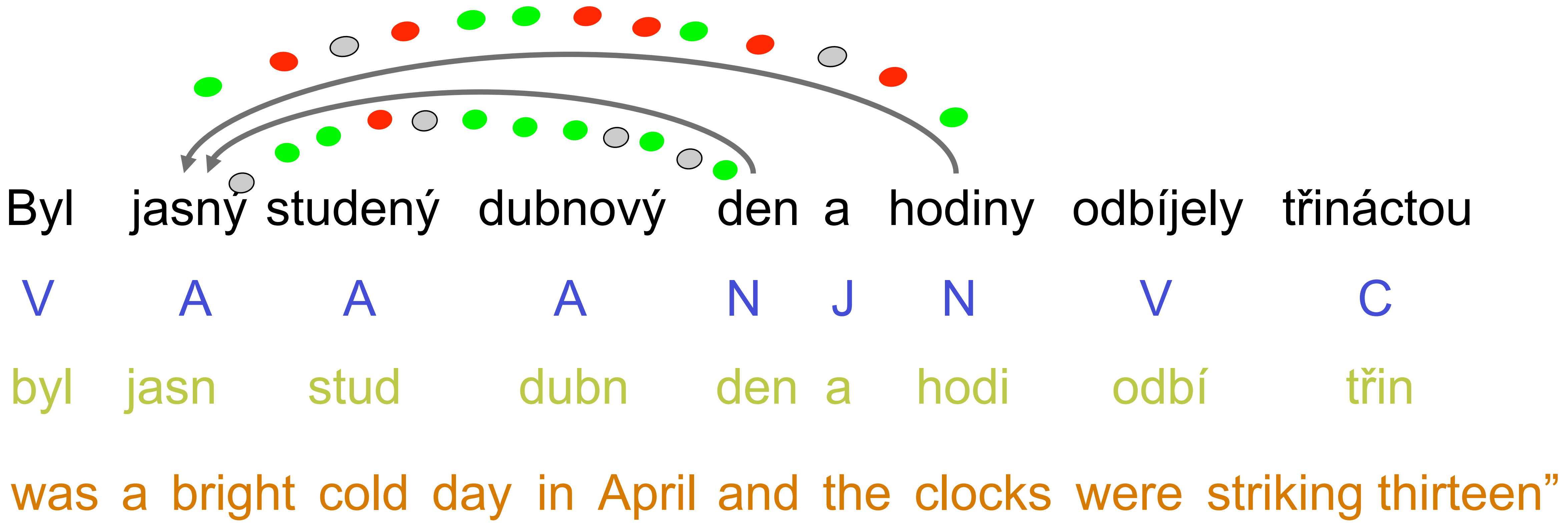
- How about this competing edge?



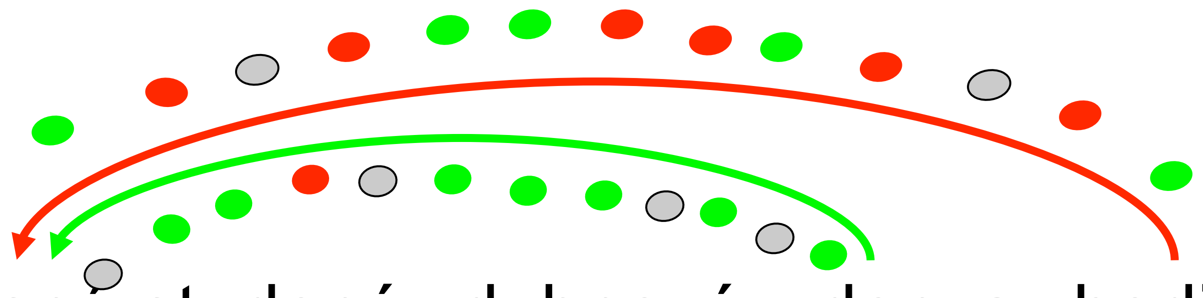
jasný ← hodiny
A ← N where N follows a conjunction
jasn ← hodi ("bright clock," stems only)
$A_{plural} \leftarrow N_{singular}$
Byl jasný studený dubnový den a hodiny odbíjely třináctou
V A A A N J N V C
byl jasn stud dubn den a hodi odbí třin
"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- 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)

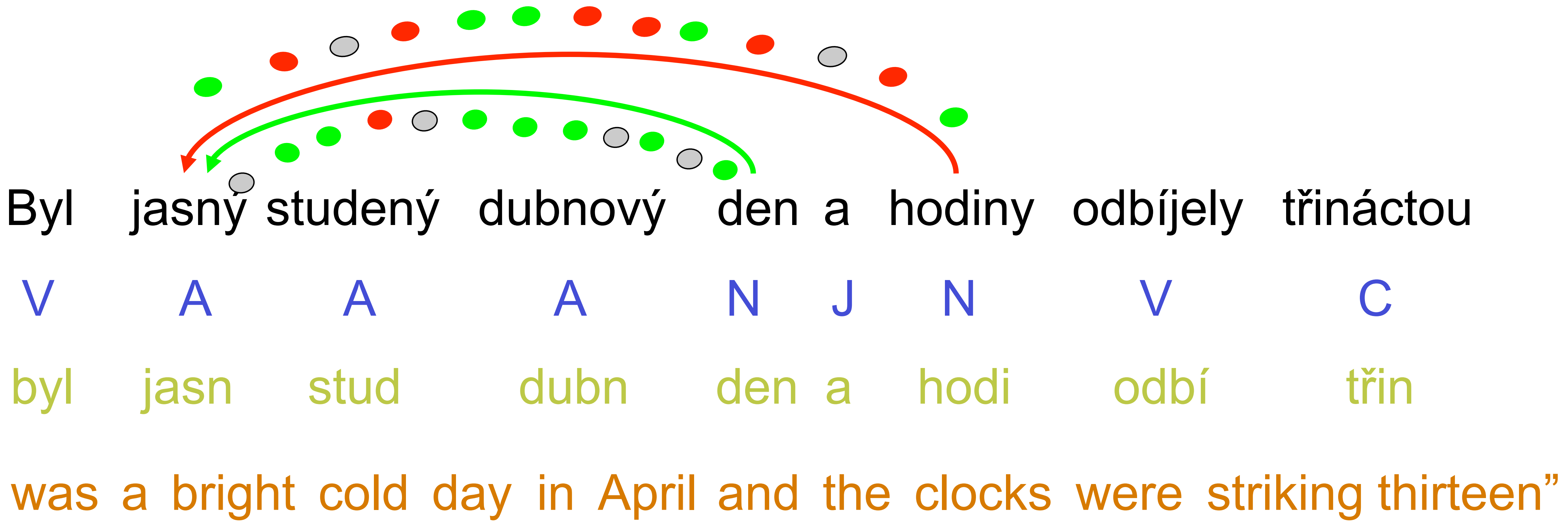


Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

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

Edge-Factored Parsers (McDonald et al. 2005)

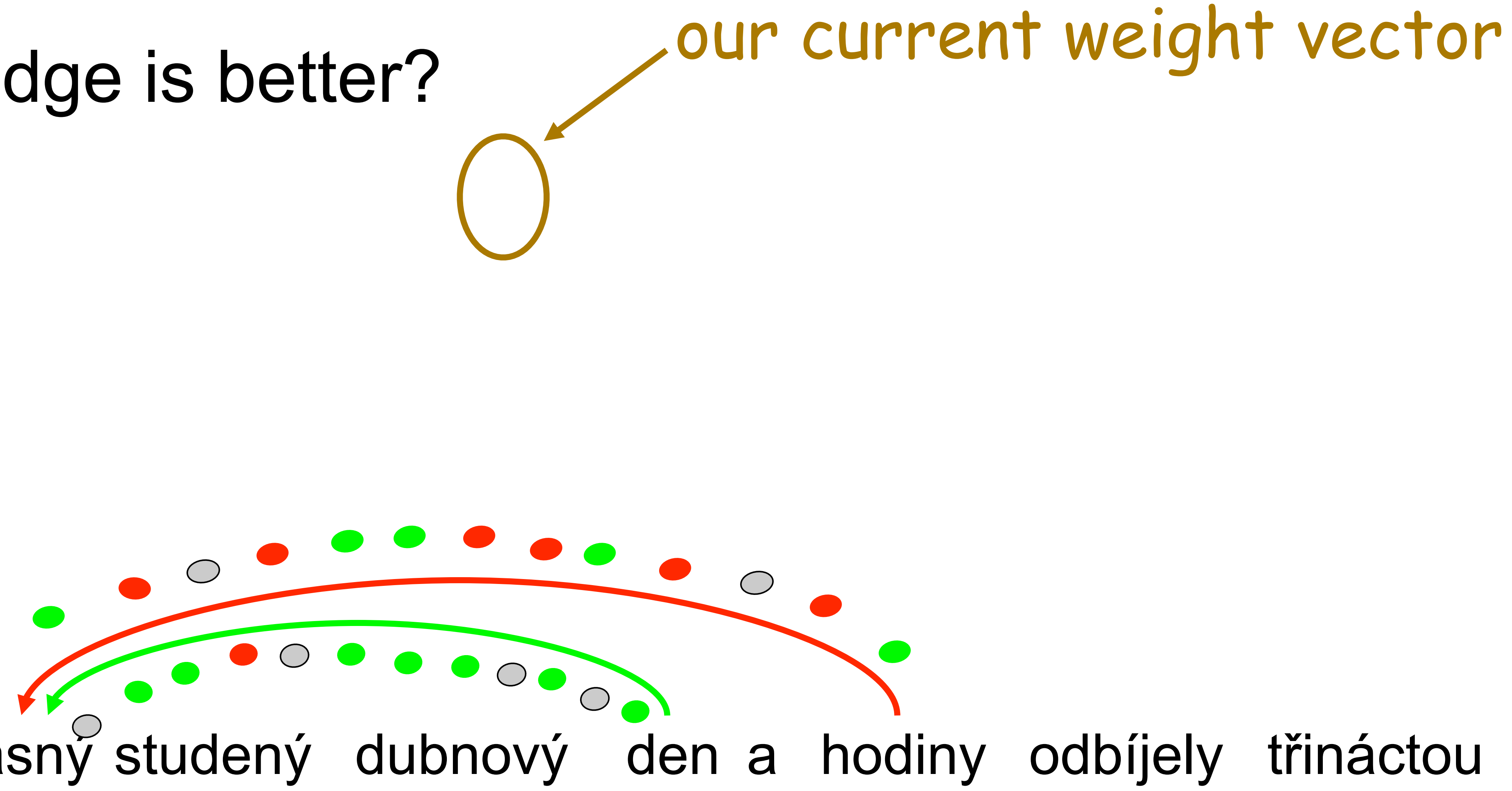
- Which edge is better?



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?



Byl jasný studený dubnový den a hodiny odbíjely třináctou

V A A A N J N V C

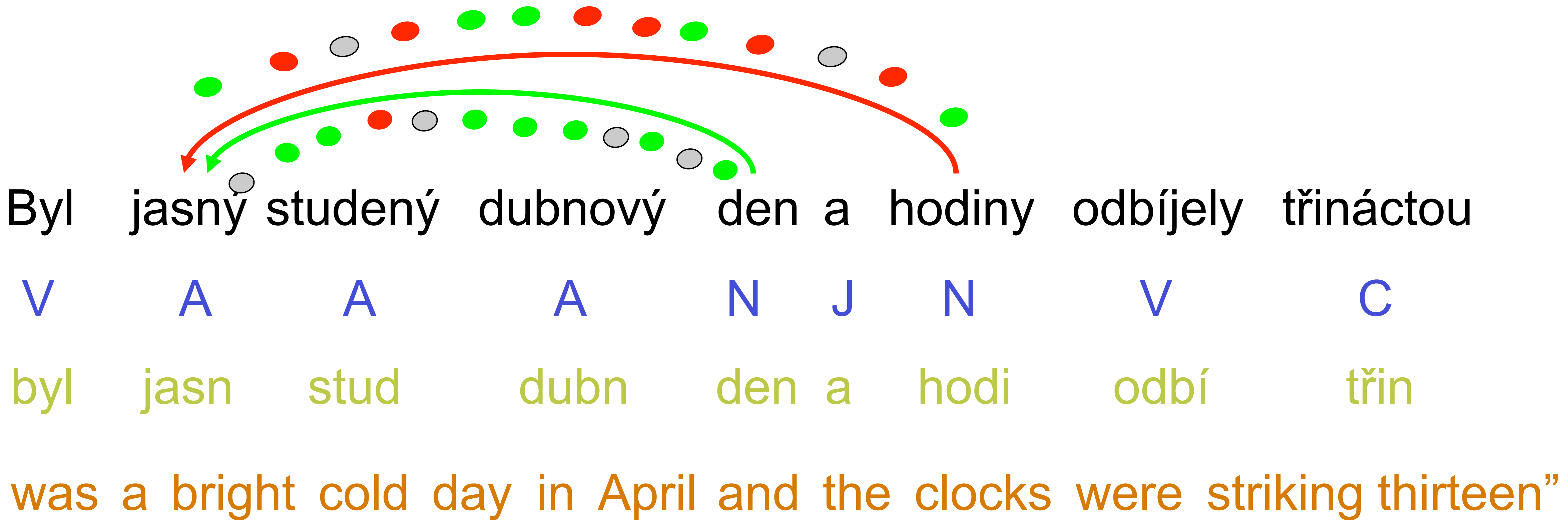
byl jasn stud dubn den a hodi odbí třin

"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?
- Score of an edge e = θ · **features**(e)

our current weight vector

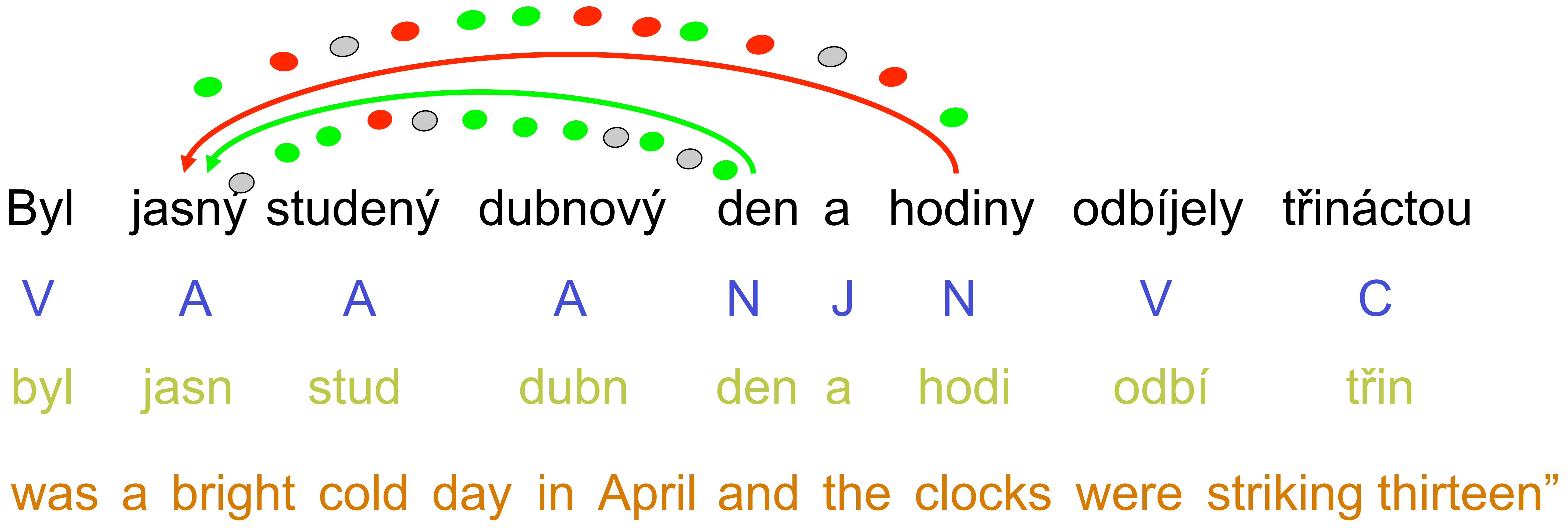


Byl jasný studený dubnový den a hodiny odbíjely třináctou
V A A A N J N V C
byl jasn stud dubn den a hodi odbí třin
"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

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

our current weight vector


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

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 total score

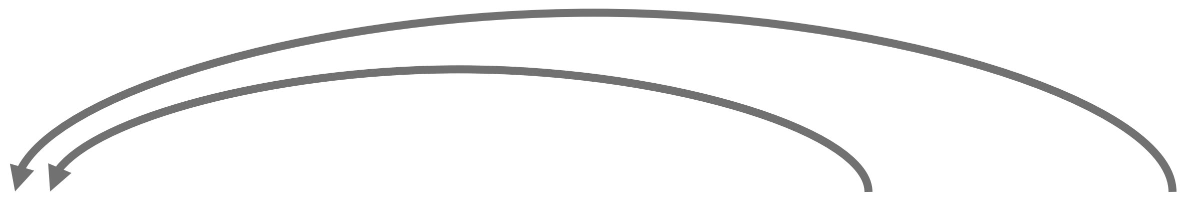
can't have both

(one parent per word)

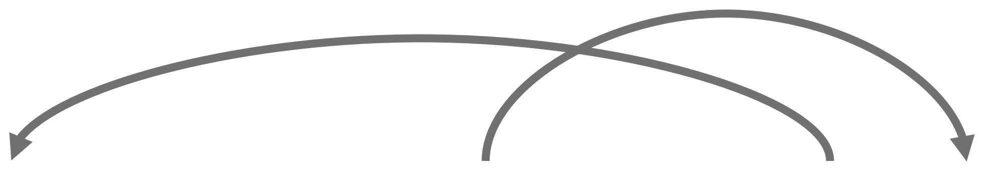
Edge-Factored Parsers (McDonald et al. 2005)

our current weight vector

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



can't have both
(one parent per word)

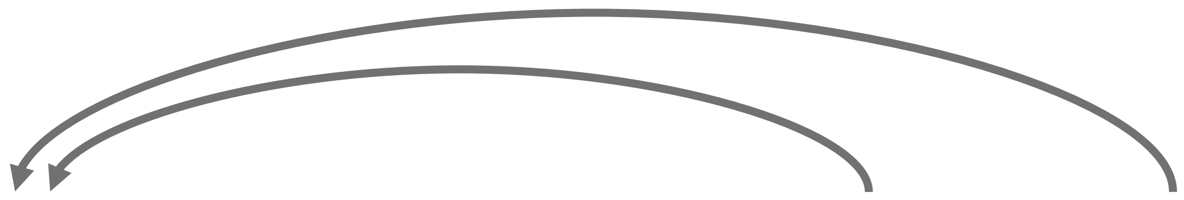


can't have both
(no crossing links)

Edge-Factored Parsers (McDonald et al. 2005)

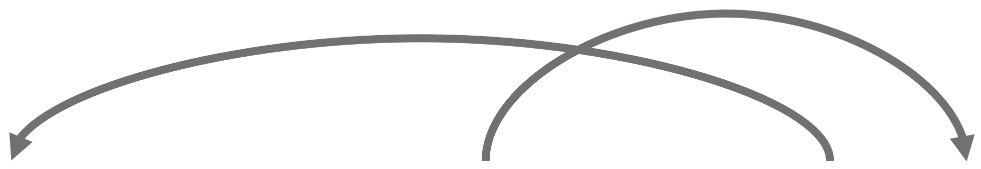
our current weight vector

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



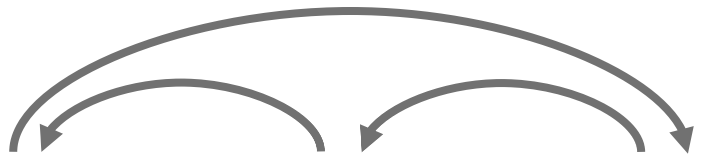
can't have both

(one parent per word)



can't have both

(no crossing links)



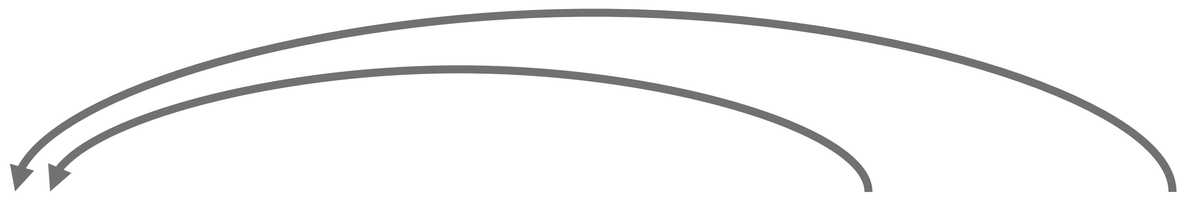
Can't have all three

(no cycles)

Edge-Factored Parsers (McDonald et al. 2005)

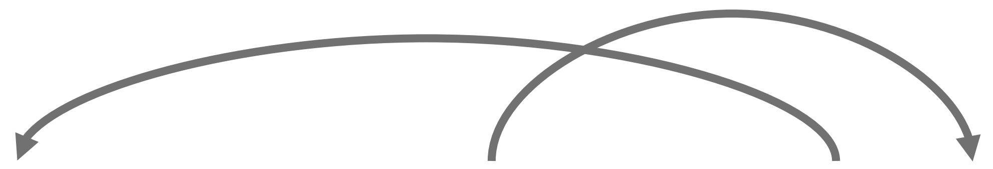
our current weight vector

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



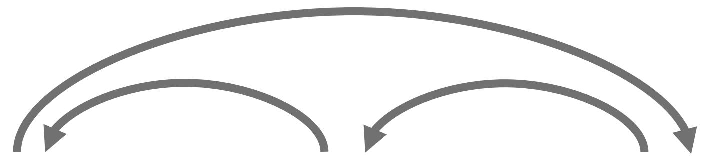
can't have both

(one parent per word)



can't have both

(no crossing links)



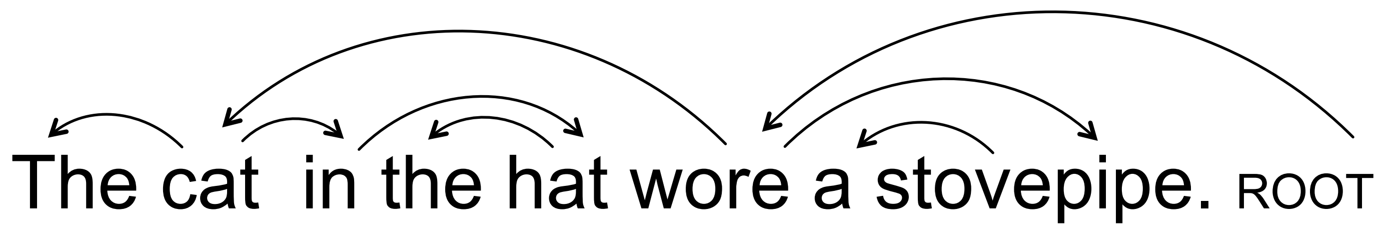
Can't have all three

(no cycles)

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

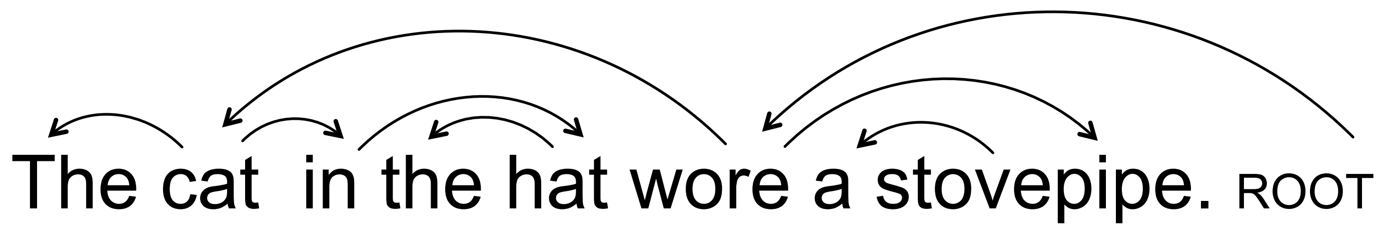
Finding Highest-Scoring Parse

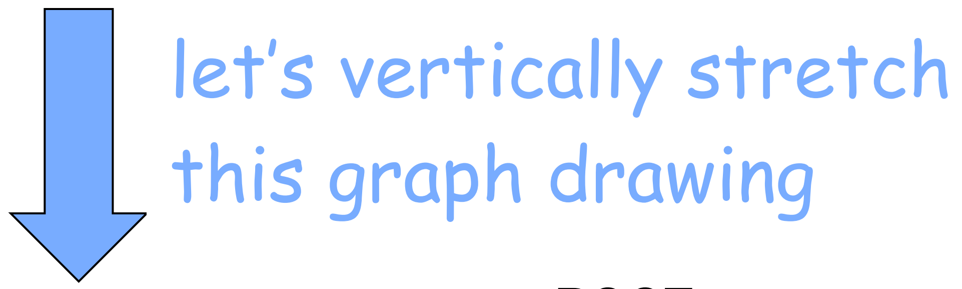
- Convert to context-free grammar (CFG)
- Then use dynamic programming

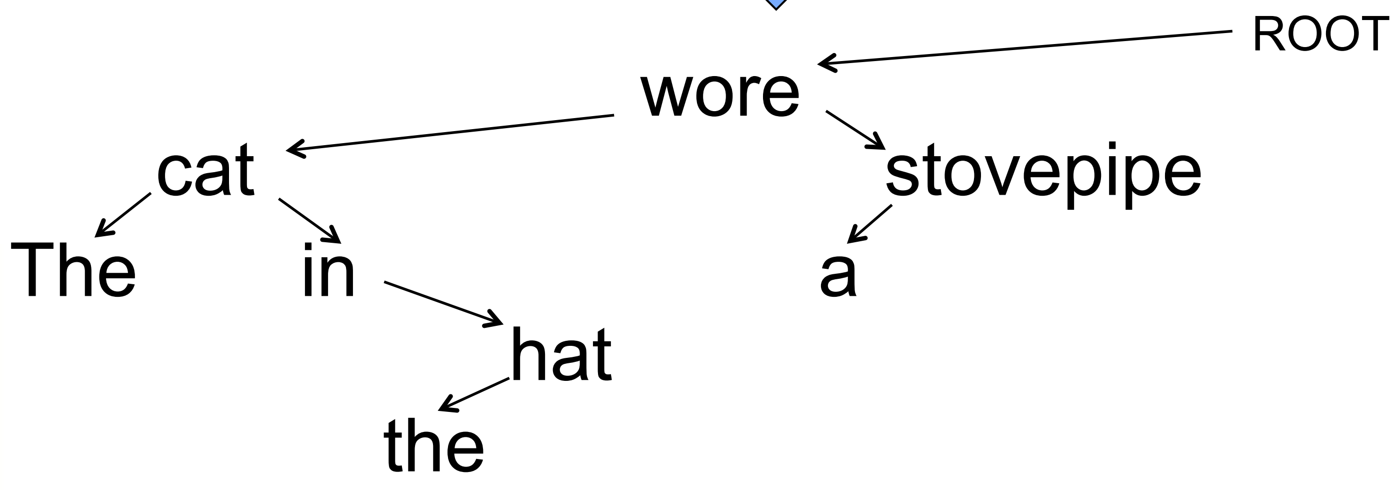


Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming

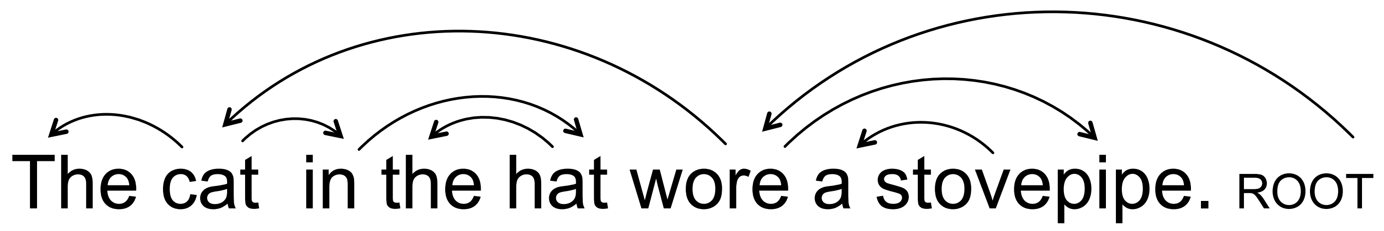


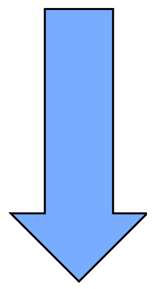




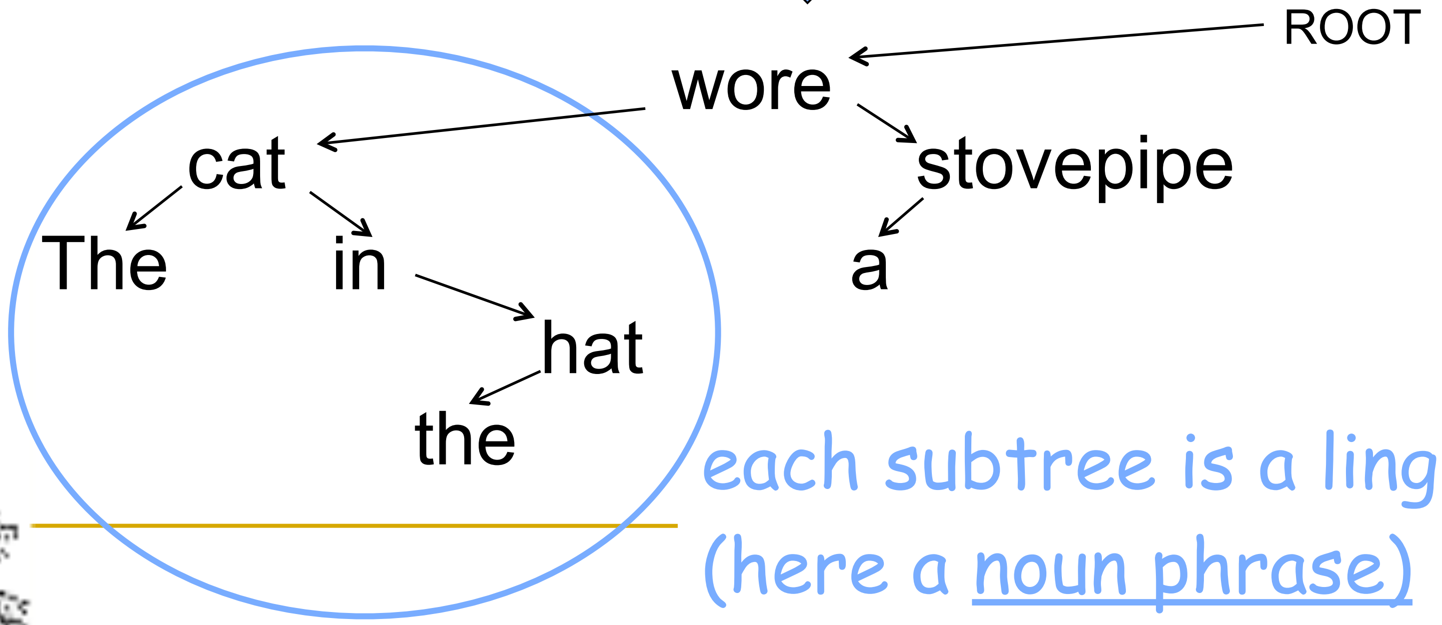
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming





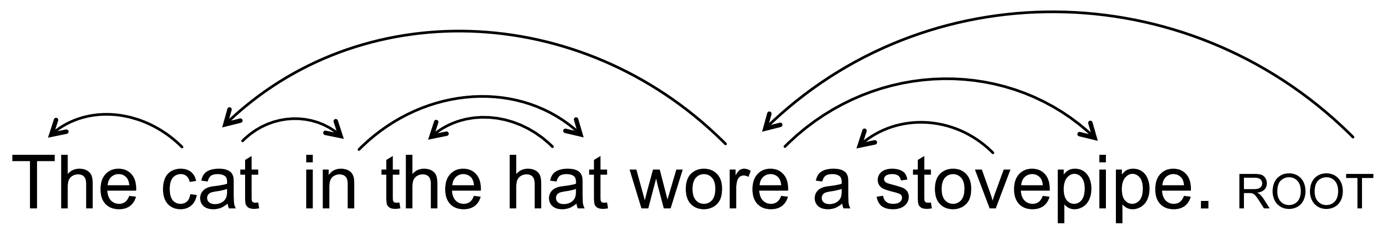
let's vertically stretch this graph drawing

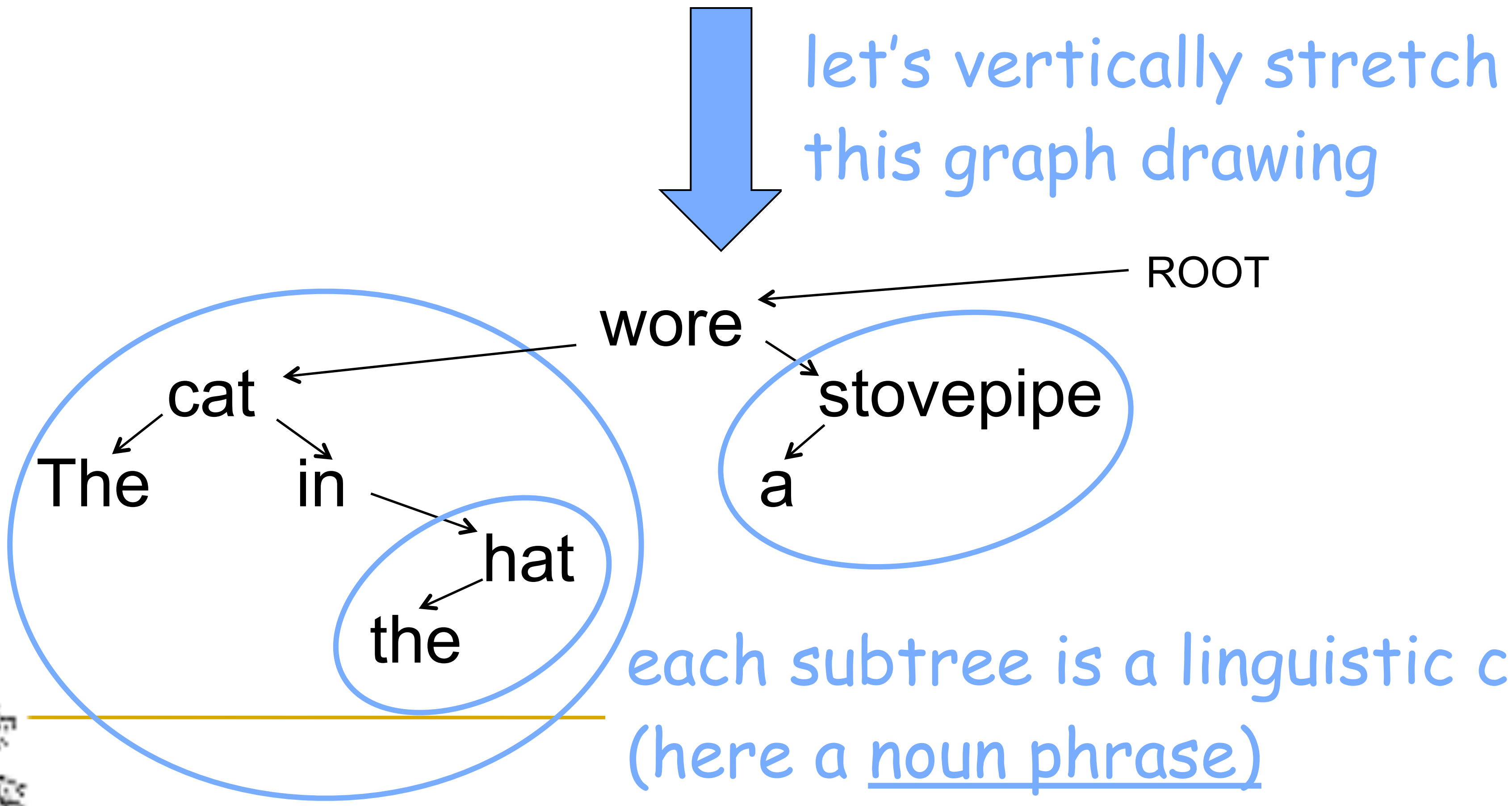


each subtree is a linguistic constituent (here a noun phrase)

Finding Highest-Scoring Parse

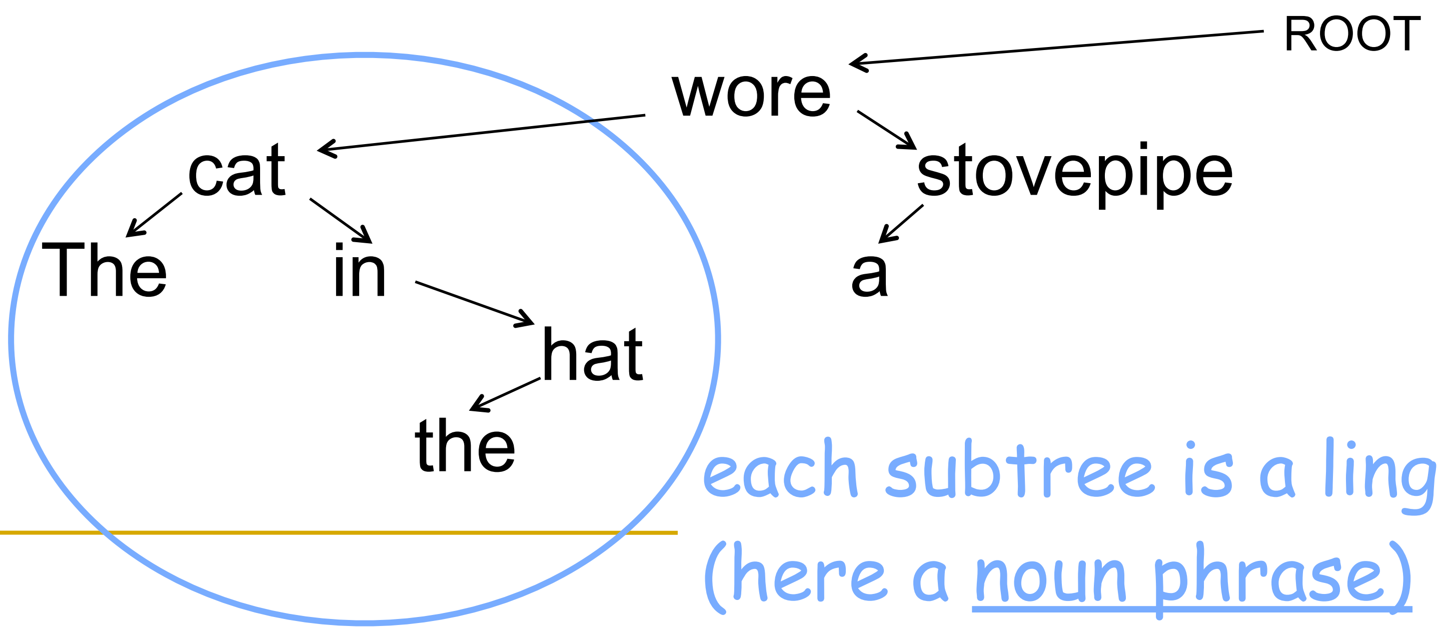
- Convert to context-free grammar (CFG)
- Then use dynamic programming





each subtree is a linguistic constituent (here a noun phrase)

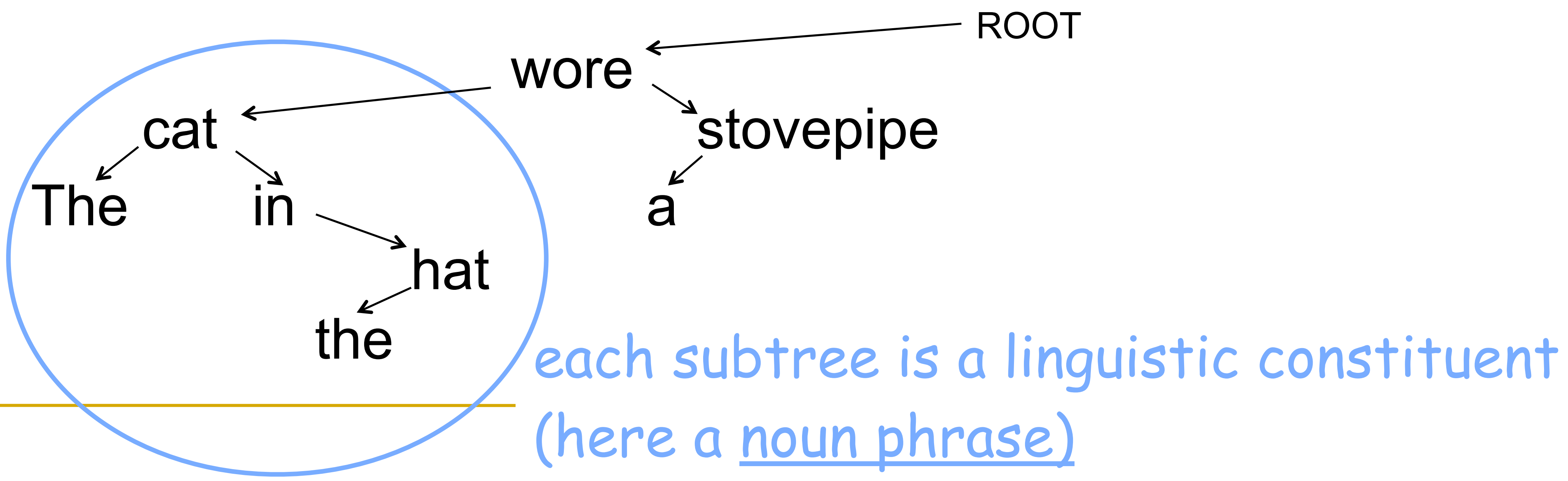
Finding Highest-Scoring Parse



each subtree is a linguistic constituent
(here a noun phrase)

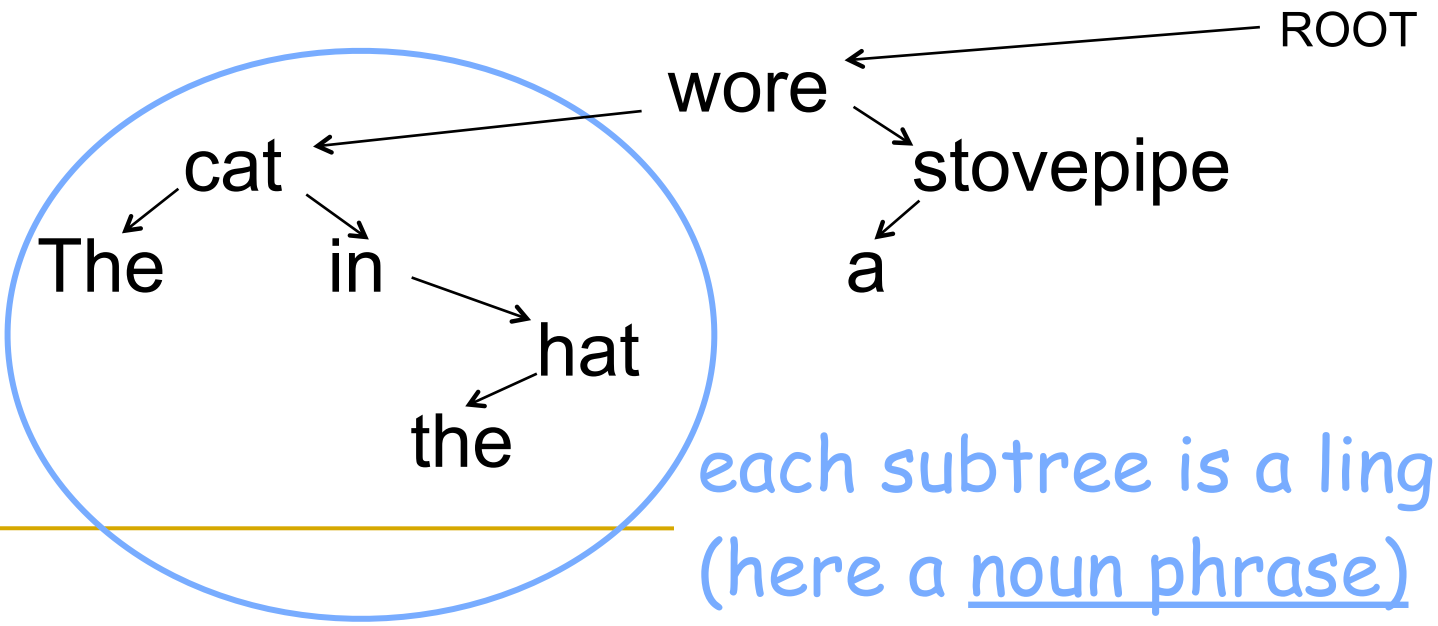
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$



Finding Highest-Scoring Parse

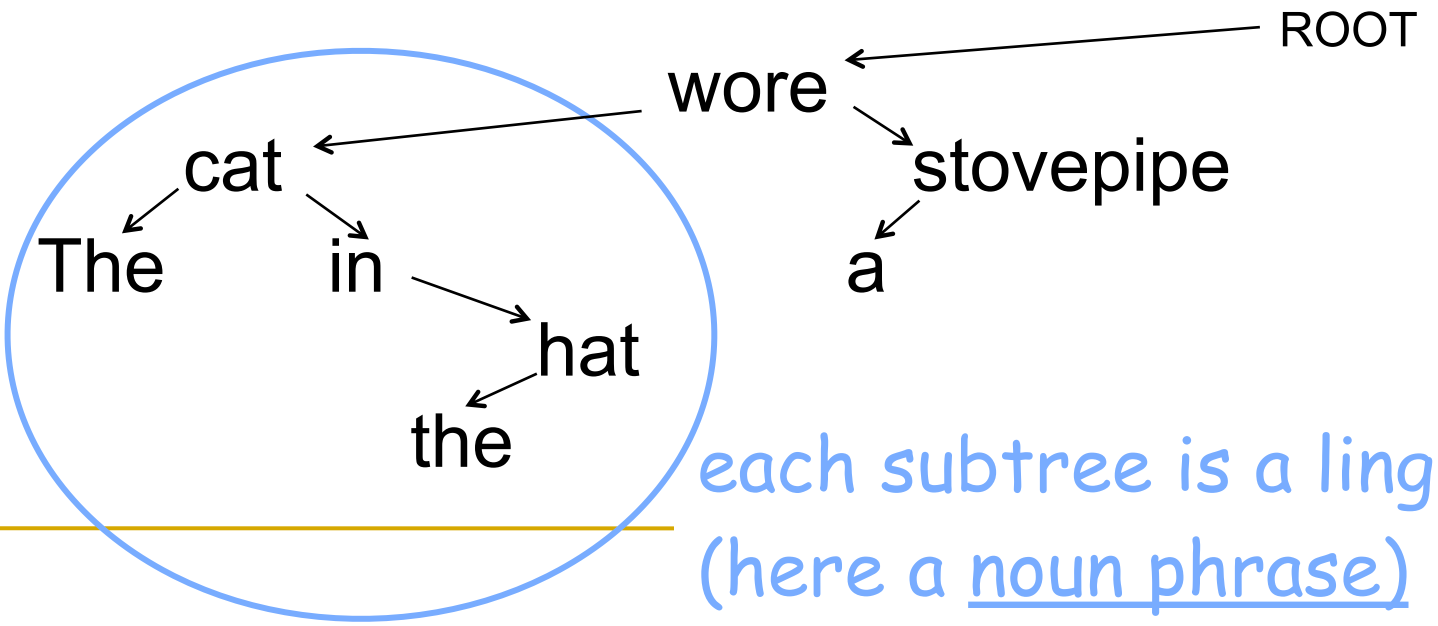
- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case



each subtree is a linguistic constituent (here a noun phrase)

Finding Highest-Scoring Parse

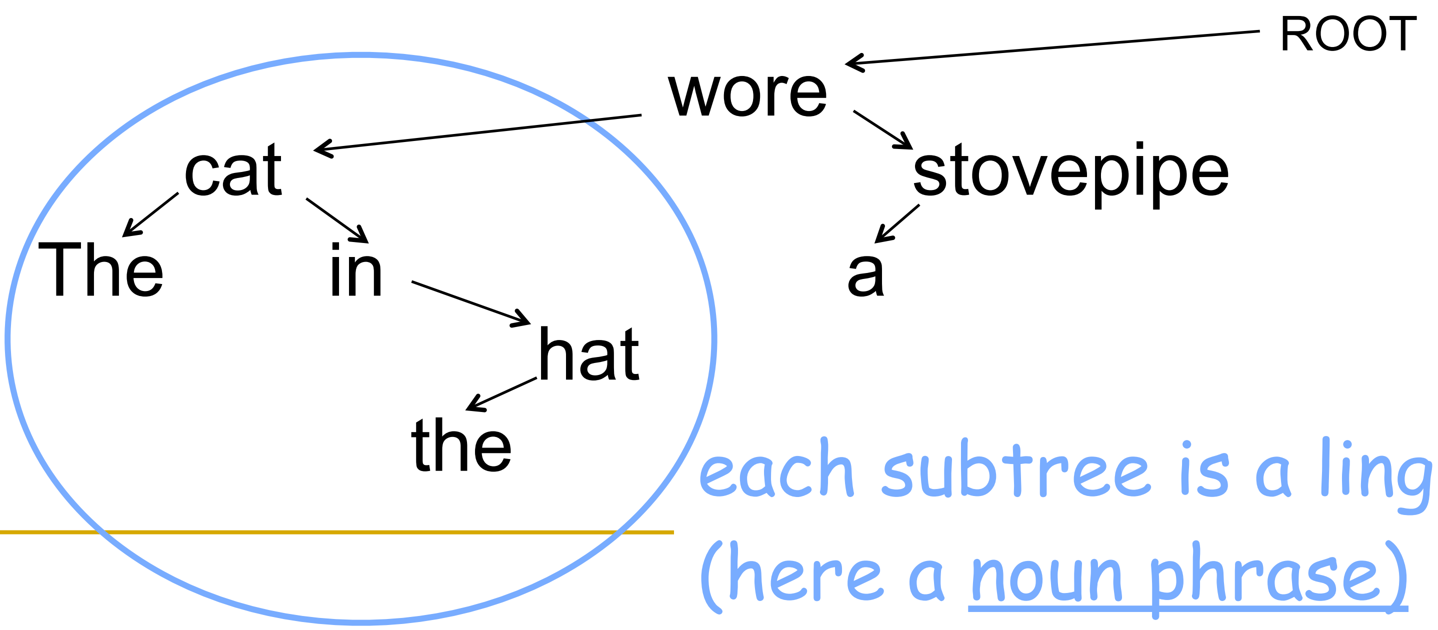
- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
    - to score "cat ← wore" link, not enough to know this is NP



each subtree is a linguistic constituent (here a noun phrase)

Finding Highest-Scoring Parse

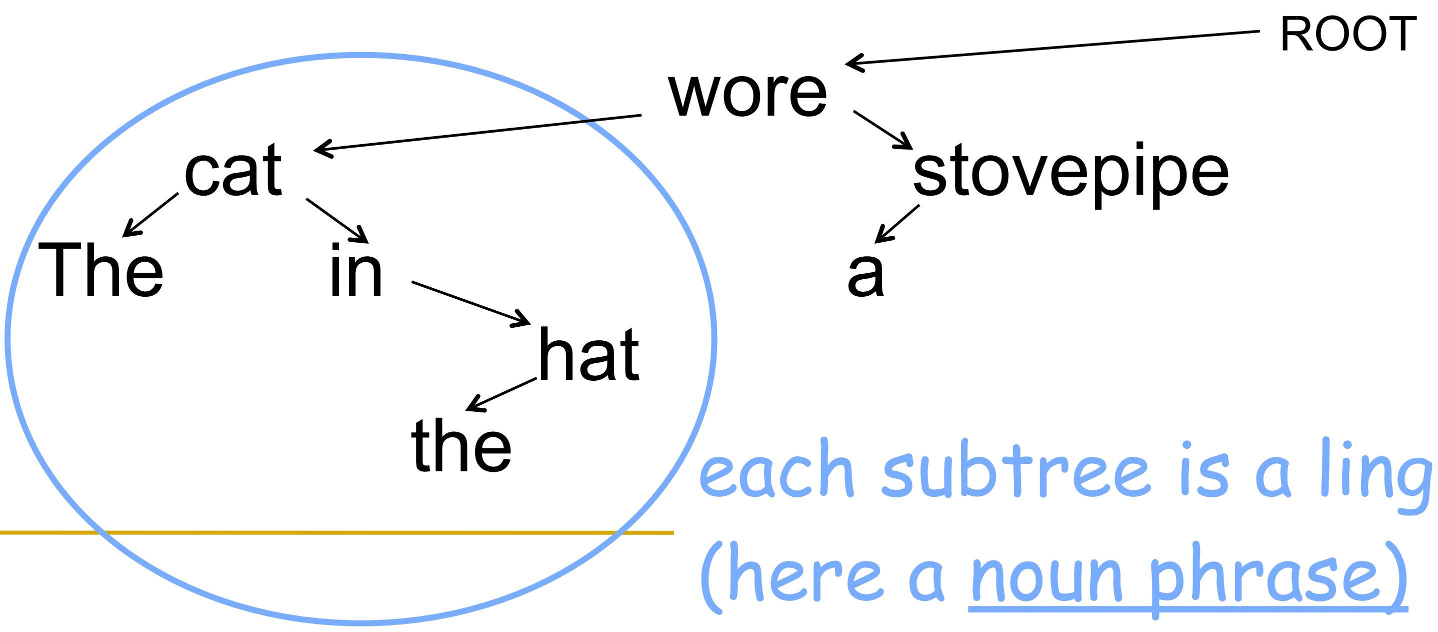
- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
    - to score "cat ← wore" link, not enough to know this is NP
    - must know it's rooted at "cat"



each subtree is a linguistic constituent (here a noun phrase)

Finding Highest-Scoring Parse

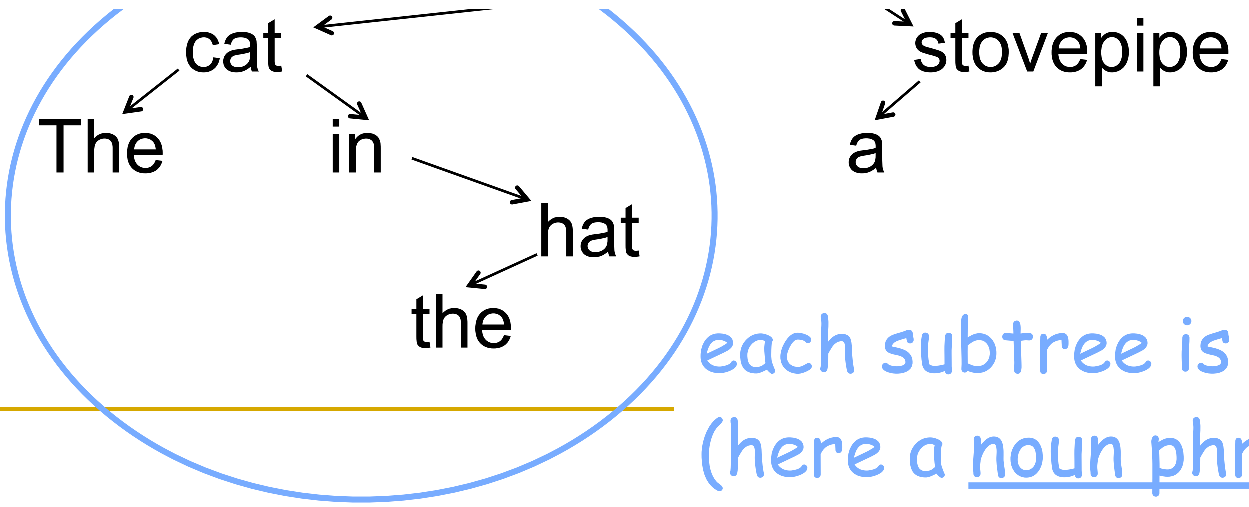
- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
    - to score "cat ← wore" link, not enough to know this is NP
    - must know it's rooted at "cat"
    - so expand nonterminal set by O(n): {NP_{the}, NP_{cat}, NP_{hat}, ...}



each subtree is a linguistic constituent (here a noun phrase)

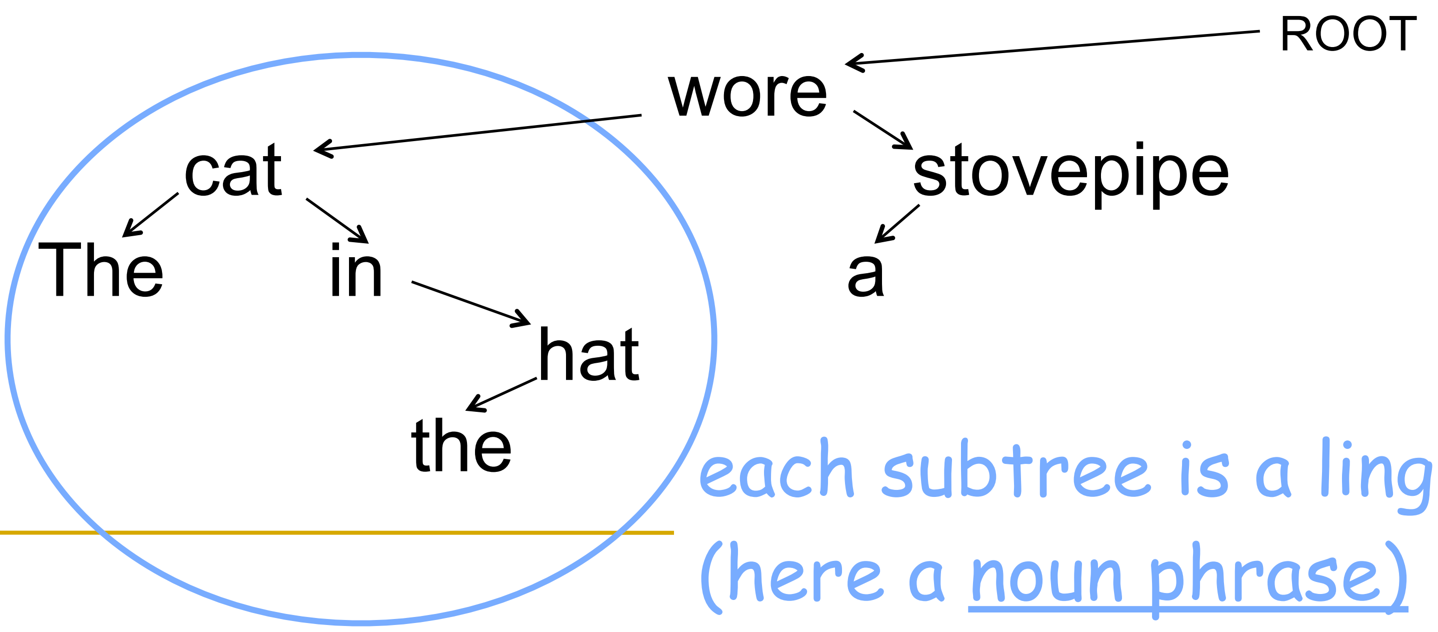
Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
    - to score "cat ← wore" link, not enough to know this is NP
    - must know it's rooted at "cat"
    - so expand nonterminal set by O(n): $\{NP_{the}, NP_{cat}, NP_{hat}, ...\}$
    - so CKY's "grammar constant" is no longer constant ☹



each subtree is a linguistic constituent (here a noun phrase)

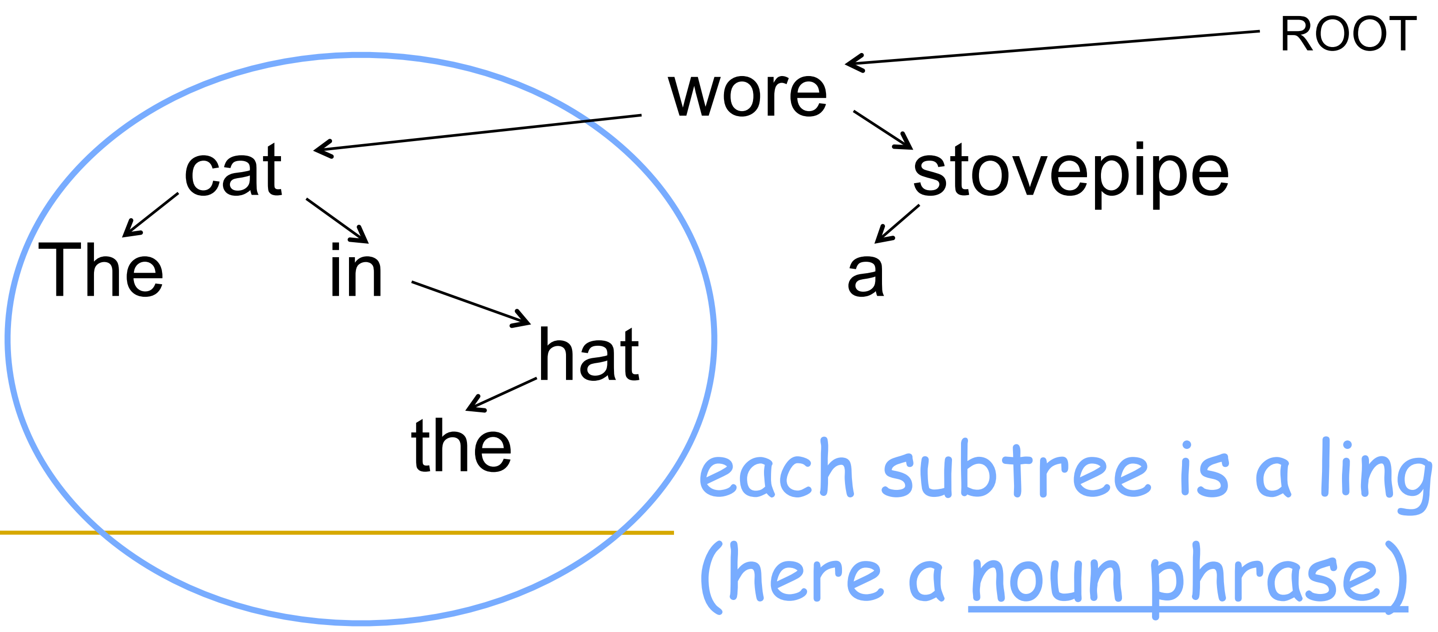
Finding Highest-Scoring Parse



each subtree is a linguistic constituent (here a noun phrase)

Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is O(n^3)
  - Unfortunately, O(n^5) in this case
  - Solution: Use a different decomposition (Eisner 1996)
    - Back to O(n^3)

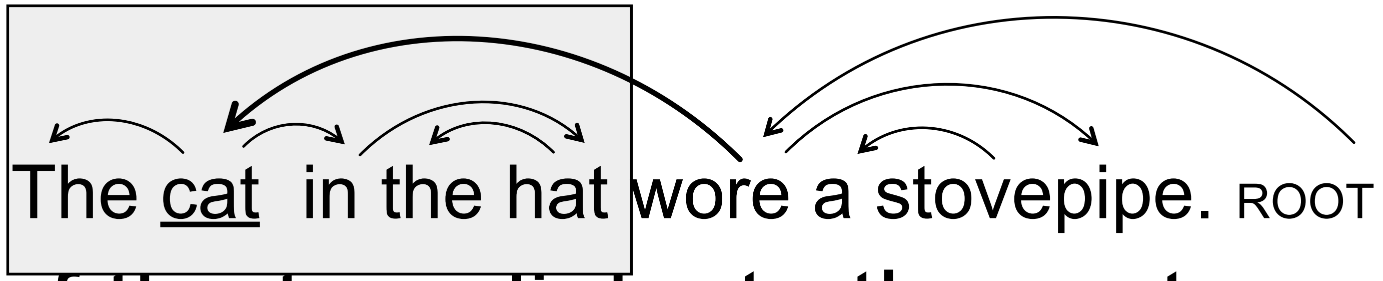


each subtree is a linguistic constituent (here a noun phrase)

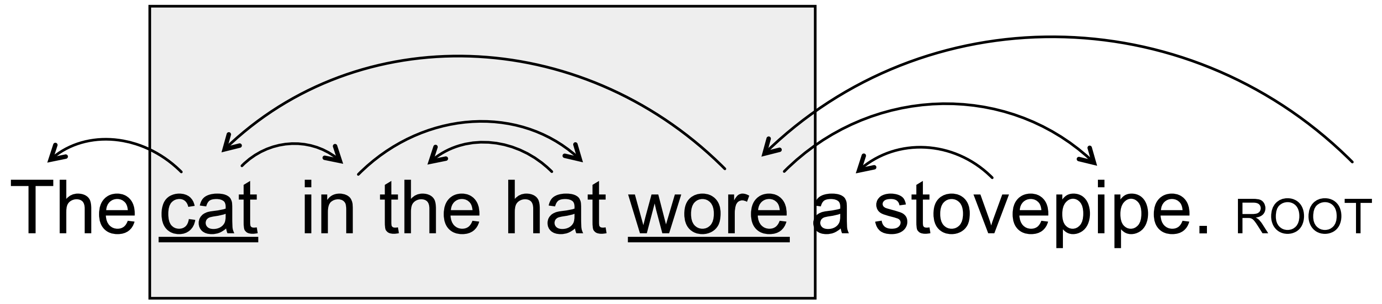
Spans vs. constituents

Two kinds of substring.

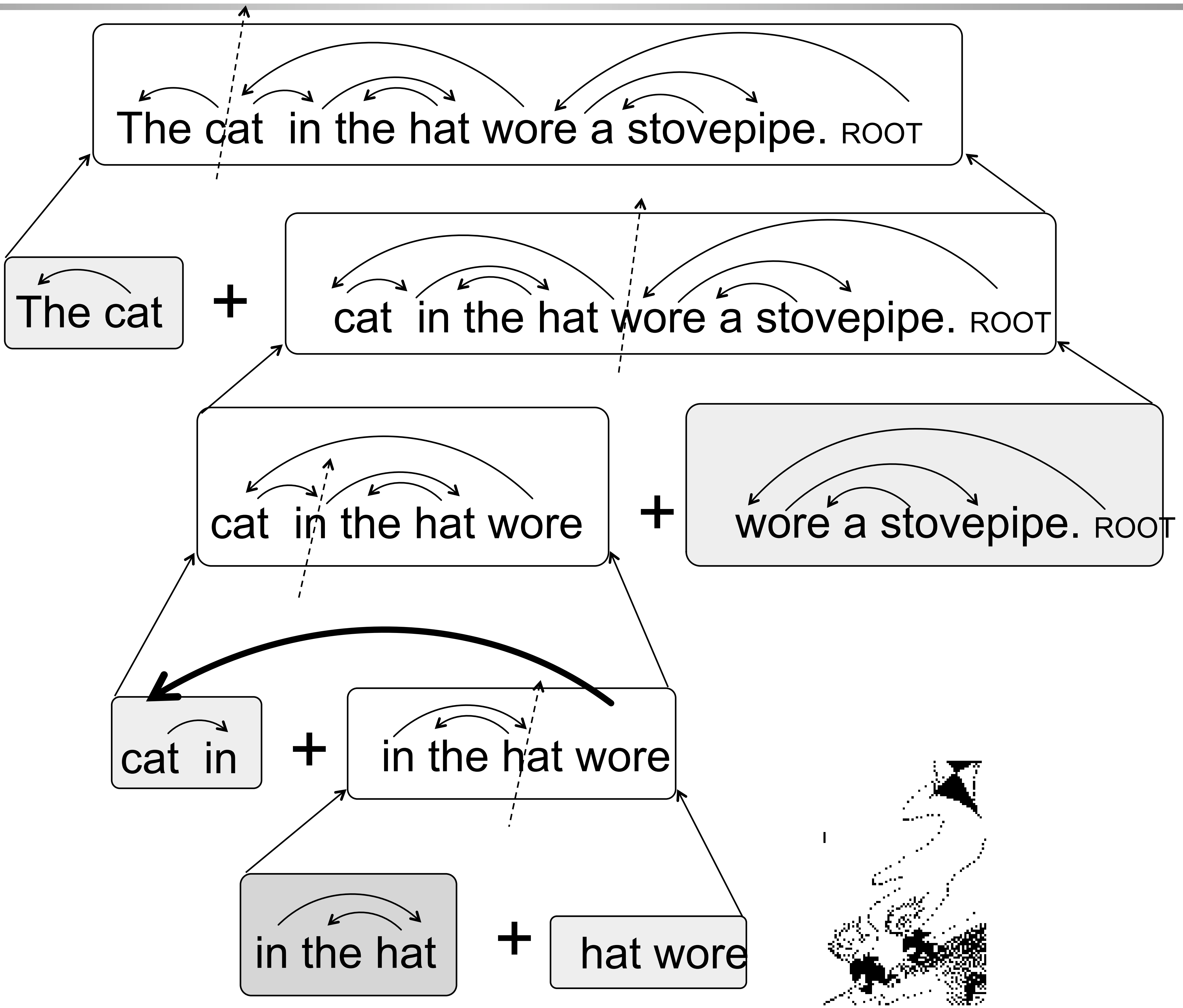
» **<u>Constituent</u>** of the tree: links to the rest only through its <u>headword</u> (root).



» **<u>Span</u>** of the tree: links to the rest only through its <u>endwords</u>.



Decomposing a tree into spans



Finding Highest-Scoring Parse

Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
  - Solution: Use a different decomposition (Eisner 1996)
    - Back to $O(n^3)$

Finding Highest-Scoring Parse

- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is $O(n^3)$
  - Unfortunately, $O(n^5)$ in this case
  - Solution: Use a different decomposition (Eisner 1996)
    - Back to $O(n^3)$
- Can play usual tricks for dynamic programming parsing
  - Further refining the constituents or spans
    - Allow prob. model to keep track of even more internal information
  - A*, best-first, coarse-to-fine
  - Training by EM etc.

Finding Highest-Scoring Parse

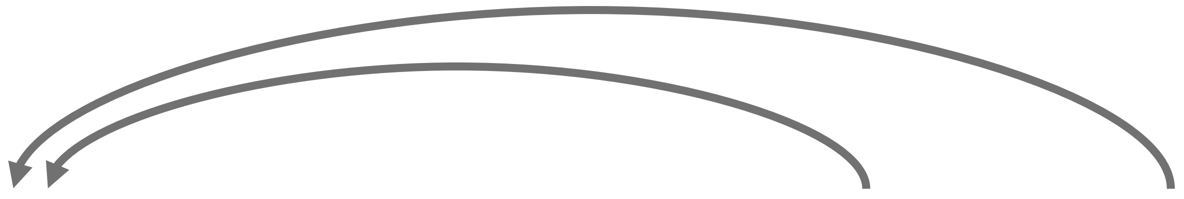
- Convert to context-free grammar (CFG)
- Then use dynamic programming
  - CKY algorithm for CFG parsing is O(n^3)
  - Unfortunately, O(n^5) in this case
  - Solution: Use a different decomposition (Eisner 1996)
    - Back to O(n^3)
- Can play usual tricks for dynamic programming parsing
  - Further refining the constituents or spans
    - Allow prob. model to keep track of even more internal information
  - A*, best-first, coarse-to-fine
  - Training by EM etc.

} require "outside" probabilities of constituents, spans, or links

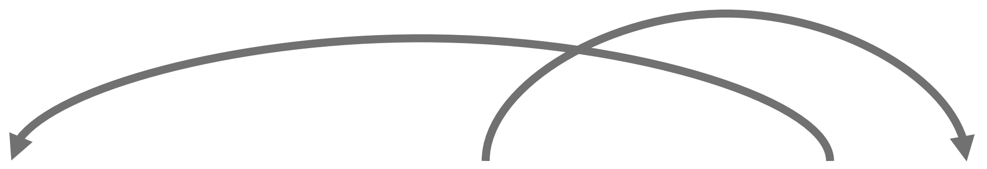
Hard Constraints on Valid Trees

our current weight vector

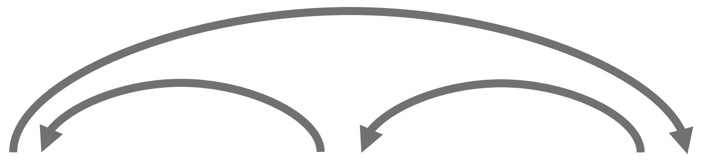
- Score of an edge e = $\theta \cdot$ **features**(e)
- Standard algos ➔ valid parse with max <u>total</u> score



can't have both
(one parent per word)



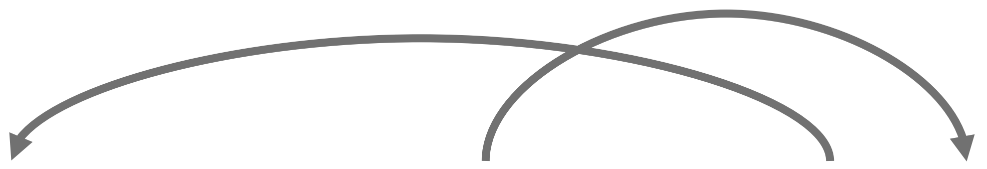
can't have both
(no crossing links)



Can't have all three
(no cycles)

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

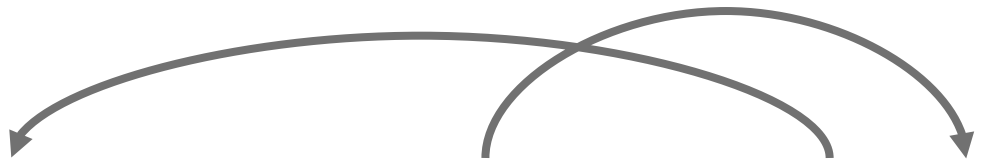
Hard Constraints on Valid Trees



can't have both

(no crossing links)

Non-Projective Parses



can't have both

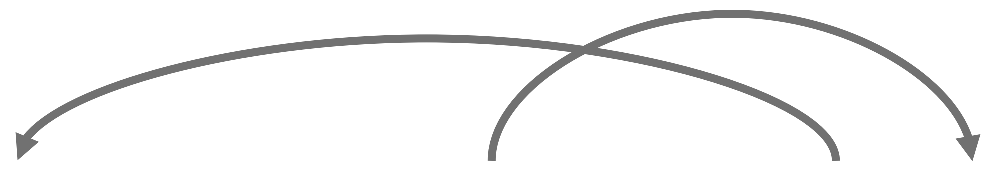
(no crossing links)

The "projectivity" restriction.

Do we really want it?

Non-Projective Parses

ROOT I 'll give a talk tomorrow on bootstrapping

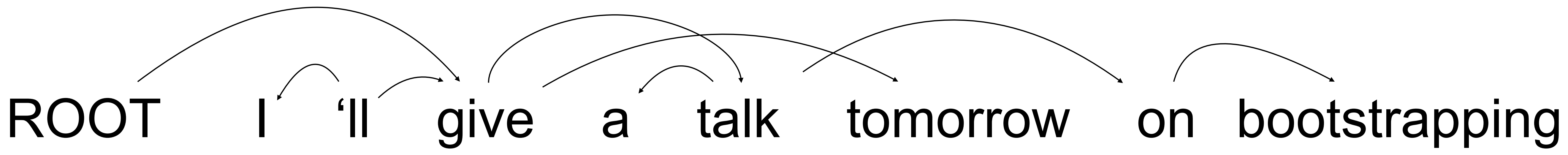


can't have both

(no crossing links)

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

Non-Projective Parses

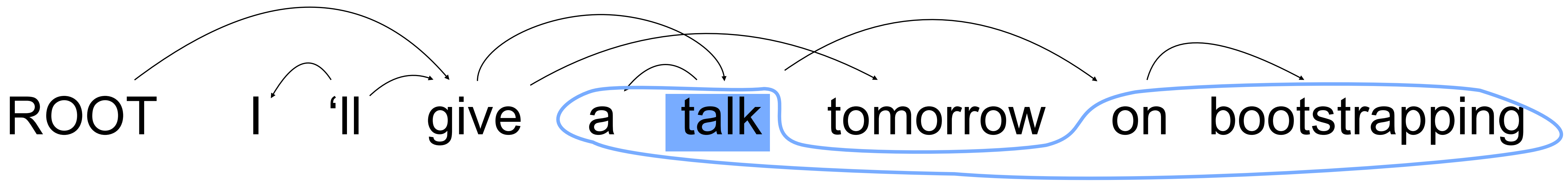


can't have both

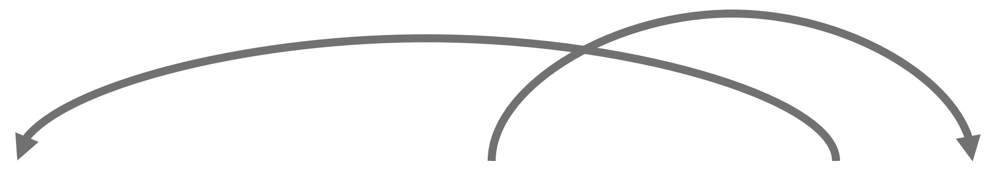
(no crossing links)

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

Non-Projective Parses



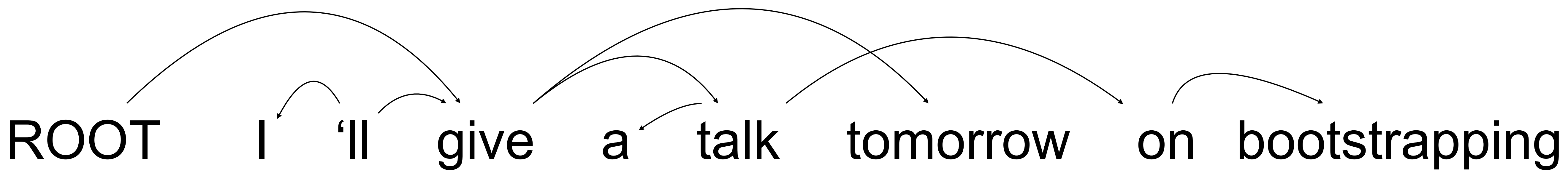
subtree rooted at "talk"
is a **discontiguous** noun phrase



can't have both
(no crossing links)

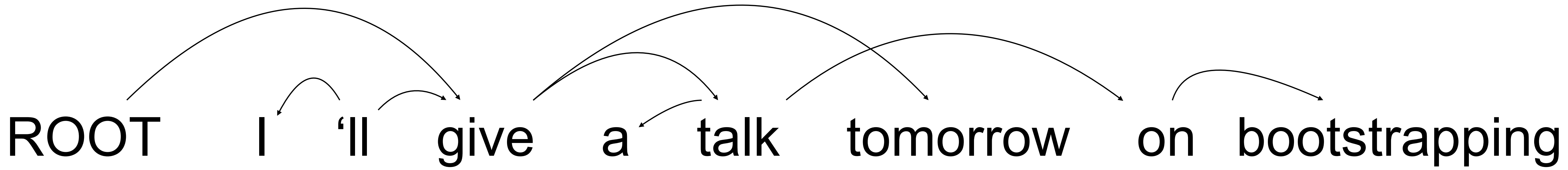
The "projectivity" restriction.
Do we really want it?

Non-Projective Parses

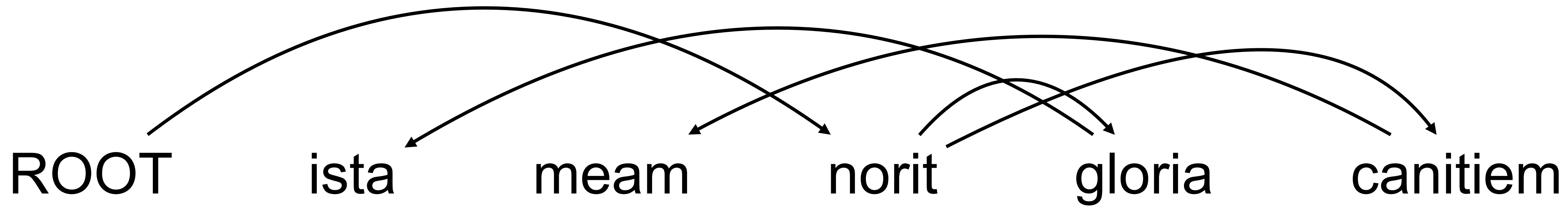


occasional non-projectivity in English

Non-Projective Parses

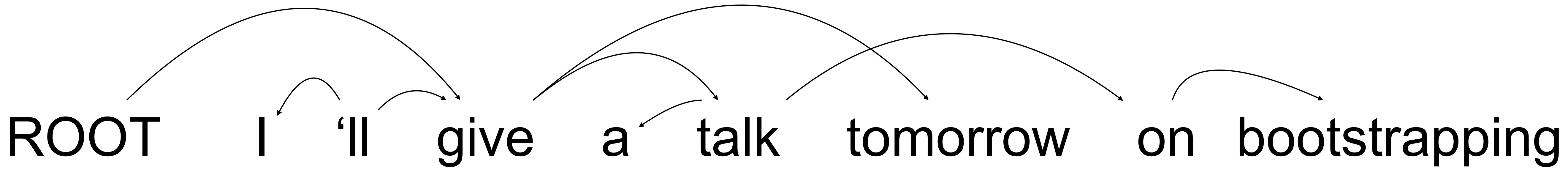


occasional non-projectivity in English

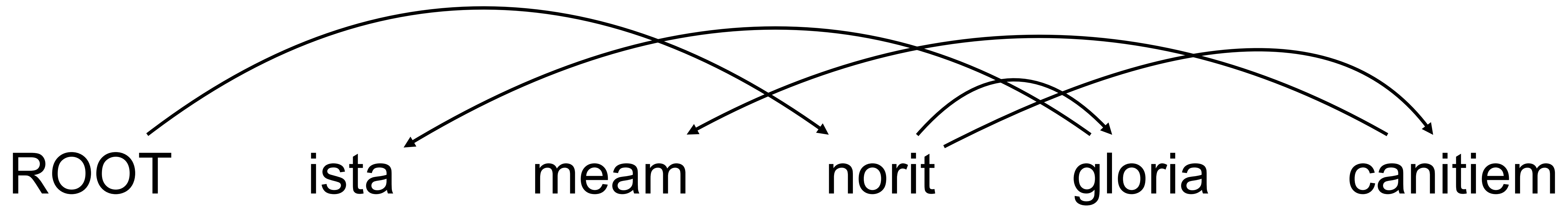


frequent non-projectivity in Latin, etc.

Non-Projective Parses



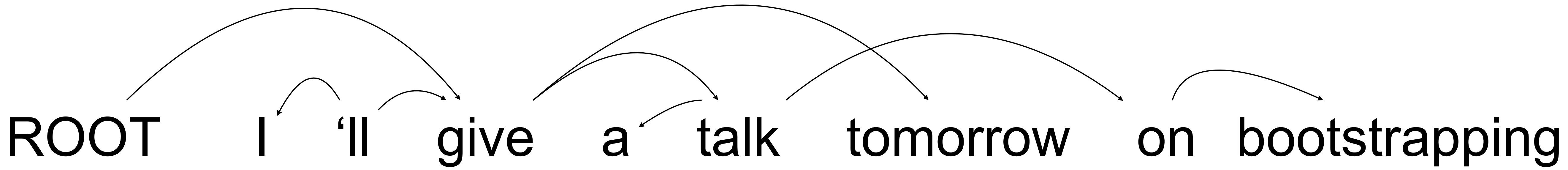
occasional non-projectivity in English



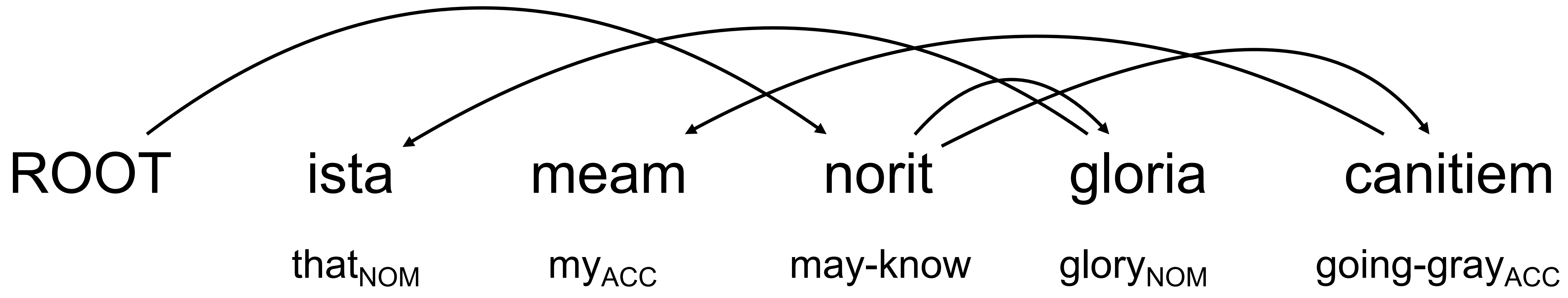
That glory may-know my going-gray
(i.e., it shall last till I go gray)

frequent non-projectivity in Latin, etc.

Non-Projective Parses



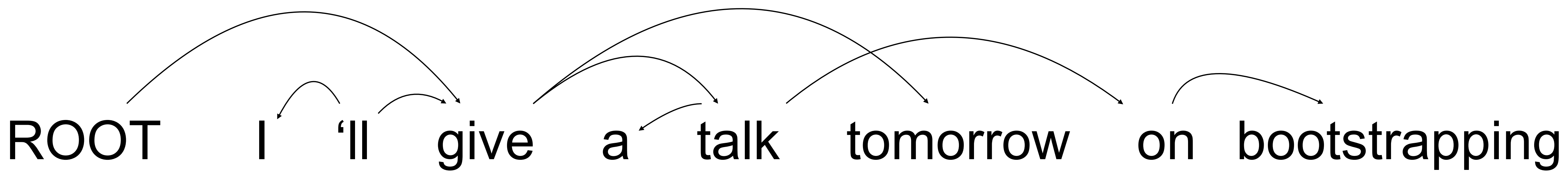
occasional non-projectivity in English



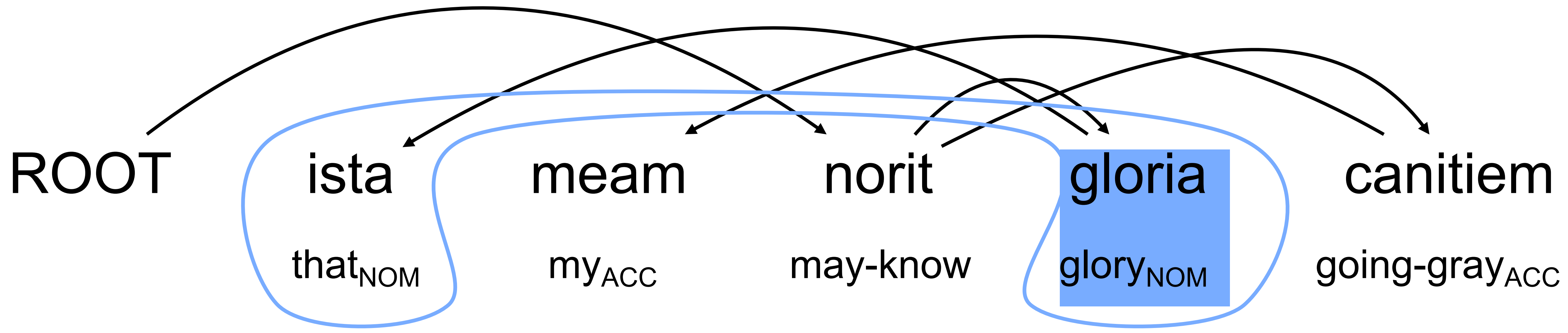
That glory may-know my going-gray
(i.e., it shall last till I go gray)

frequent non-projectivity in Latin, etc.

Non-Projective Parses



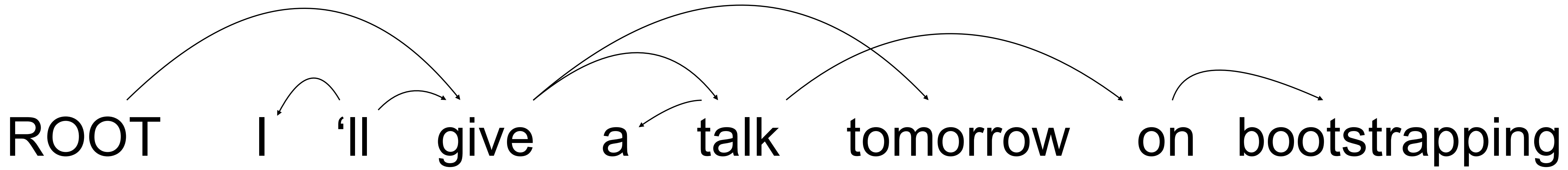
occasional non-projectivity in English



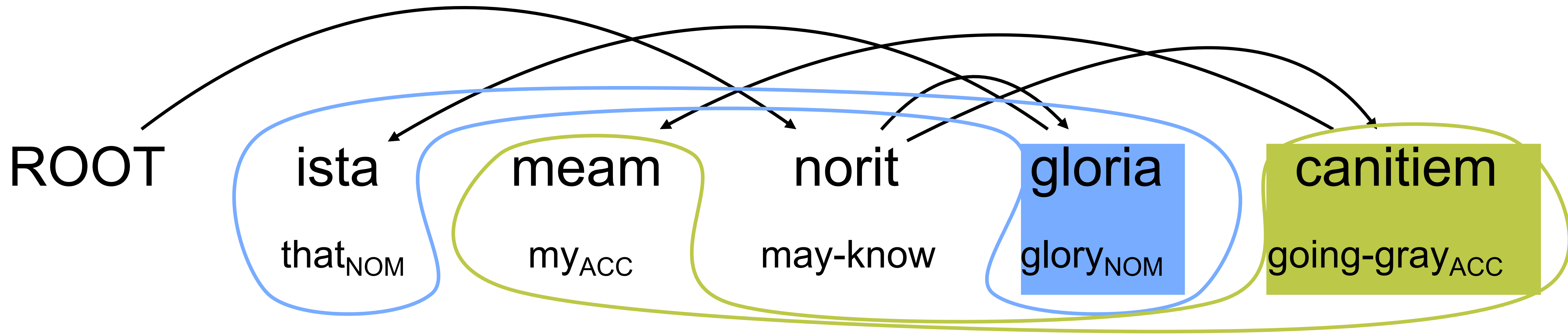
That glory may-know my going-gray
(i.e., it shall last till I go gray)

frequent non-projectivity in Latin, etc.

Non-Projective Parses



occasional non-projectivity in English

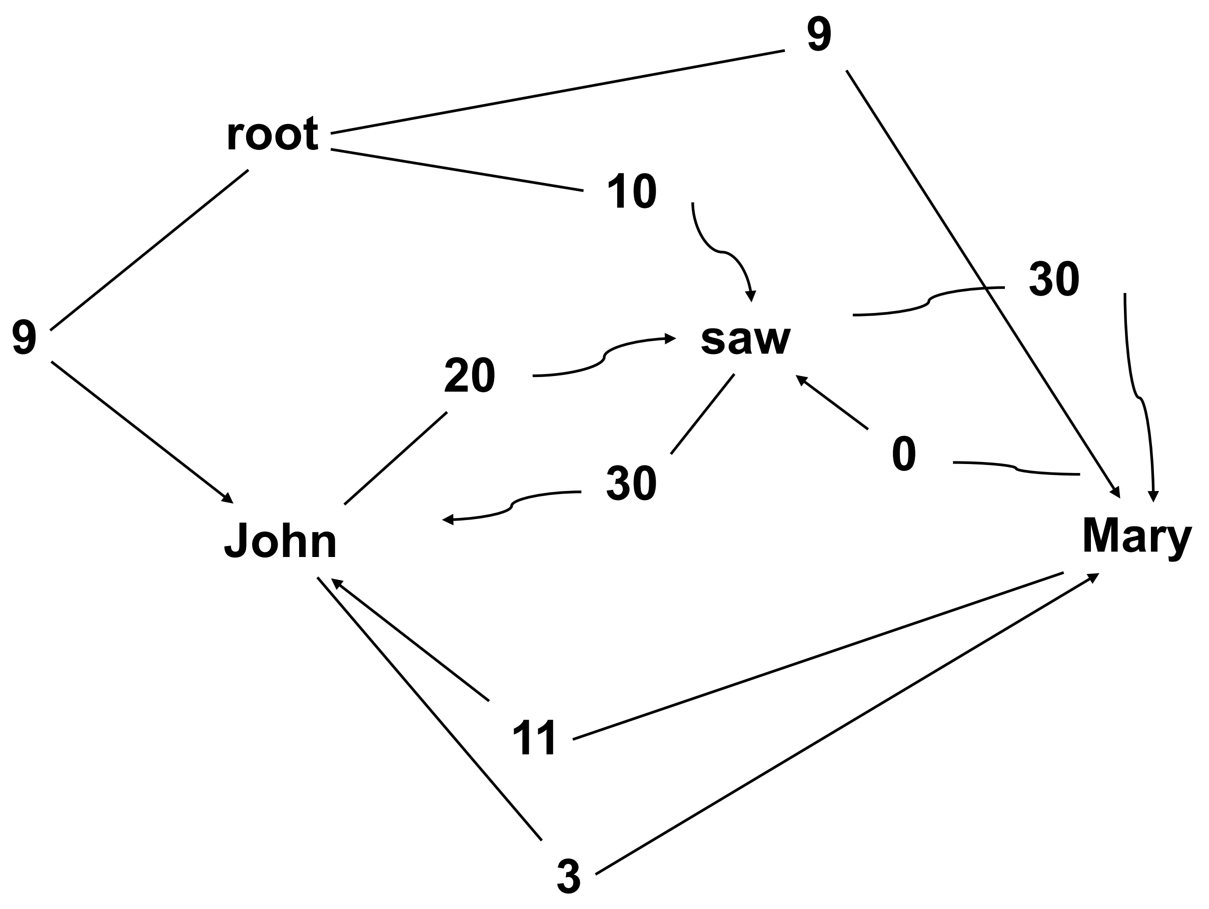


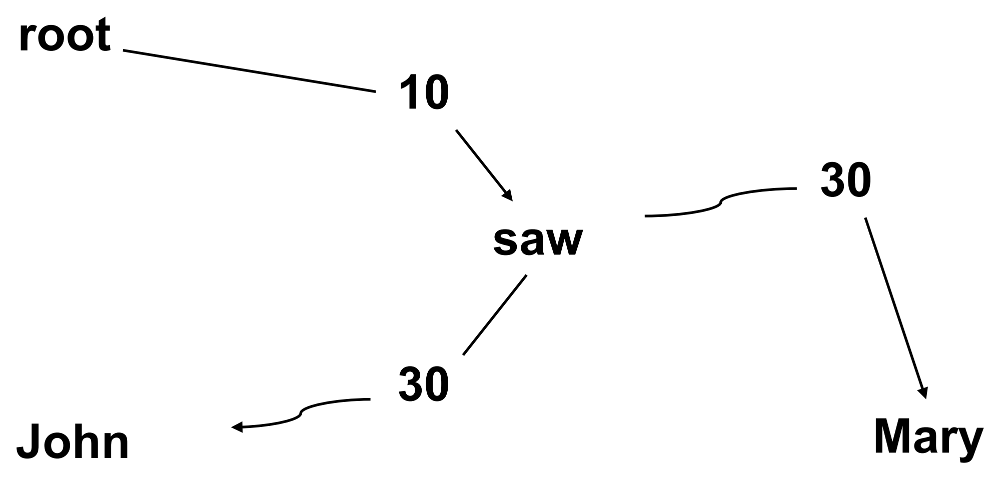
That glory may-know my going-gray
(i.e., it shall last till I go gray)

frequent non-projectivity in Latin, etc.

Finding highest-scoring non-projective tree

- 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

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 (right) – may be non-projective.
- Can be found in time $O(n^2)$.

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

slide thanks to Dragomir Radev

Graph Theory to the Rescue!

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*.

Graph Theory to the Rescue!

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!

Graph Theory to the Rescue!

$O(n^3)$ time!

T[illegible]e'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

$$\begin{bmatrix} 0 & -s(1,0) & -s(2,0) & \mathrm{L} & -s(n,0) \\ 0 & 0 & -s(2,1) & \mathrm{L} & -\ s(n,1) \\ 0 & -s(1,2) & 0 & \mathrm{L} & -s(n,2) \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & -s(1,n) & -s(2,n) & \mathrm{L} & 0 \end{bmatrix}$$

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

Building the Kirchoff (Laplacian) Matrix

$$\begin{bmatrix} 0 & -s(1,0) & -s(2,0) & \mathrm{L} & -s(n,0) \\ 0 & 0 & -s(2,1) & \mathrm{L} & -\ s(n,1) \\ 0 & -s(1,2) & 0 & \mathrm{L} & -s(n,2) \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & -s(1,n) & -s(2,n) & \mathrm{L} & 0 \end{bmatrix}$$

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

Building the Kirchoff (Laplacian) Matrix

$$\begin{bmatrix} 0 & -s(1,0) & -s(2,0) & \mathrm{L} & -s(n,0) \\ 0 & \sum_{j\neq 1} s(1,j) & -s(2,1) & \mathrm{L} & -\ s(n,1) \\ 0 & -s(1,2) & \sum_{j\neq 2} s(2,j) & \Lambda & -s(n,2) \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & -s(1,n) & -s(2,n) & \mathrm{L} & \sum_{j\neq n} s(n,j) \end{bmatrix}$$

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

Building the Kirchoff (Laplacian) Matrix

$$\begin{vmatrix} \sum_{j\neq 1} s(1,j) & -s(2,1) & \mathbf{L} & -s(n,1) \\ -s(1,2) & \sum_{j\neq 2} s(2,j) & \mathbf{L} & -s(n,2) \\ \mathbf{M} & \mathbf{M} & \mathbf{O} & \mathbf{M} \\ -s(1,n) & -s(2,n) & \mathbf{L} & \sum_{j\neq n} s(n,j) \end{vmatrix}$$

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

Building the Kirchoff (Laplacian) Matrix

$$\begin{vmatrix} \sum_{j\neq 1} s(1,j) & -s(2,1) & \mathbf{L} & -s(n,1) \\ -s(1,2) & \sum_{j\neq 2} s(2,j) & \mathbf{L} & -s(n,2) \\ \mathbf{M} & \mathbf{M} & \mathbf{O} & \mathbf{M} \\ -s(1,n) & -s(2,n) & \mathbf{L} & \sum_{j\neq n} s(n,j) \end{vmatrix}$$

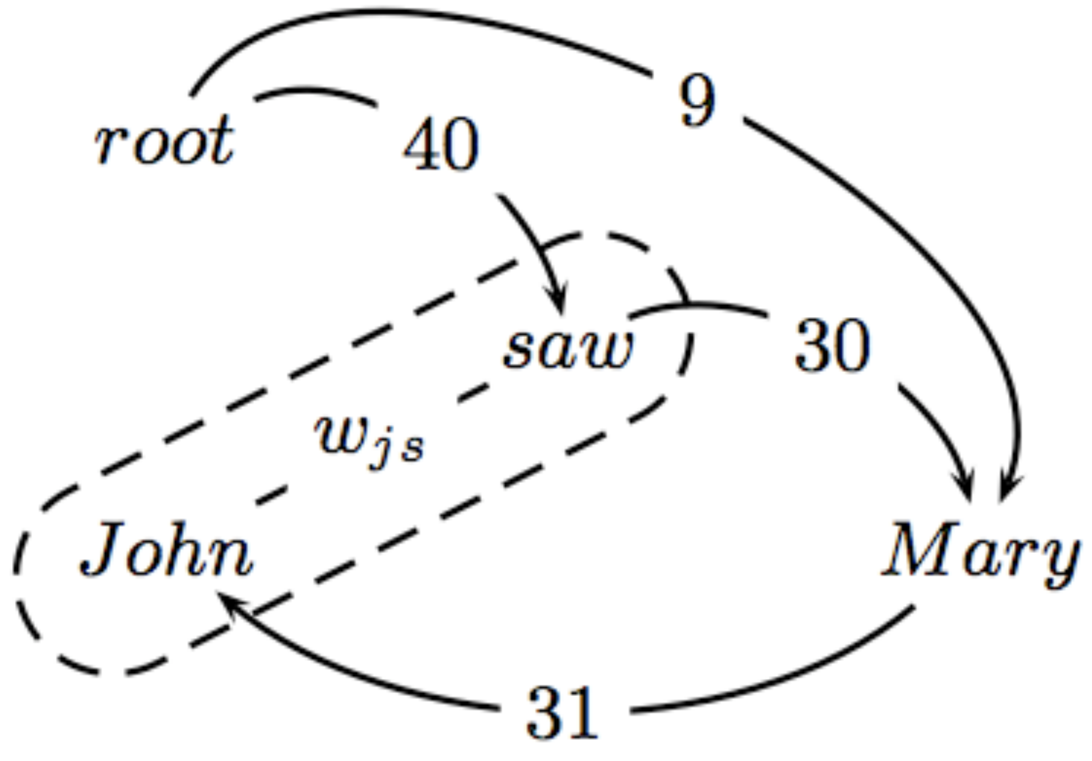
- Negate edge scores
- Sum columns (children)
- Strike root row/col.
- Take determinant

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

Why Should This Work?

Clear for 1x1 matrix; use induction

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



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

$K'' \equiv K(\{1,2\} \mid \{1,2\})$

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

Undirected case; special root cases for directed

Why Should This Work?

Clear for 1x1 matrix; use induction

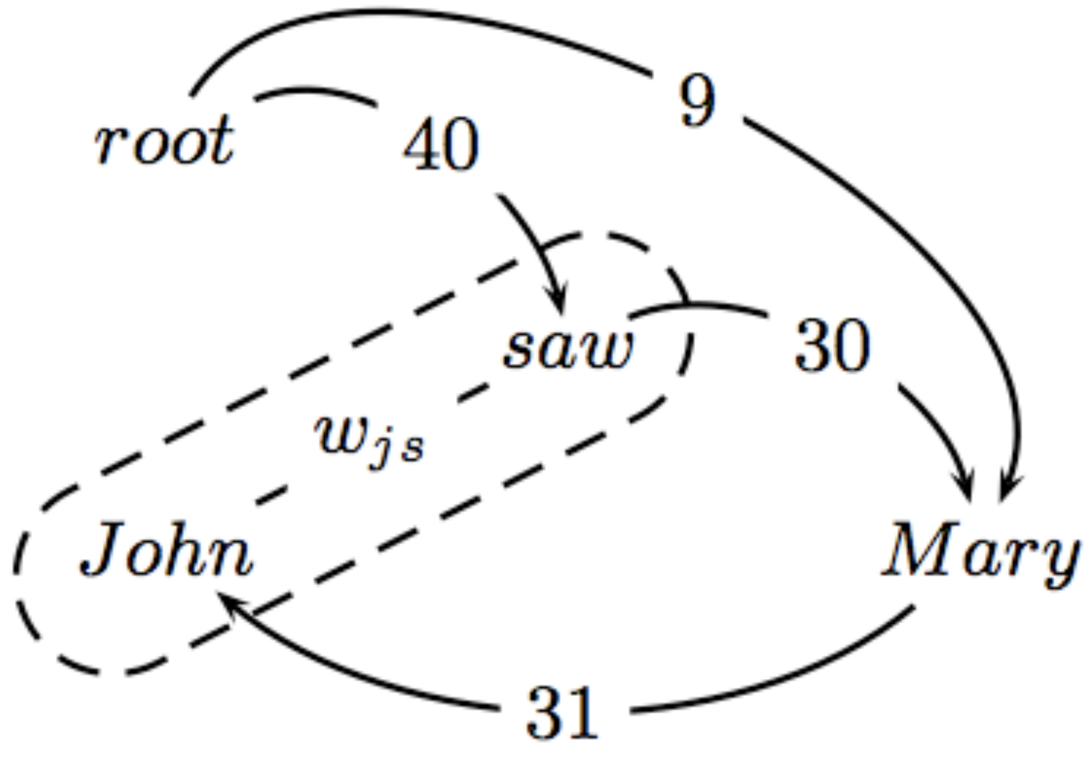
$$\begin{vmatrix} \sum_{j\neq 1} s(1,j) & -s(2,1) & \text{L} & -s(n,1) \\ -s(1,2) & \sum_{j\neq 2} s(2,j) & \Lambda & -s(n,2) \\ \text{M} & \text{M} & \text{O} & \text{M} \\ -s(1,n) & -s(2,n) & \text{L} & \sum_{j\neq n} s(n,j) \end{vmatrix}$$

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

$K'' \equiv K(\{1,2\} \mid \{1,2\})$

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

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



Undirected case; special root cases for directed