Paper Presentation & Summary

- 15% of total grade
- Presentation (3%):
  - 12-minute talk (template here) + 3-minute Q&A
  - You will be punished (-3%) if you sign up but don't present
- Paper summary (12%):
  - For each topic, submitting summaries of at least two papers (2 * 6 classes = 12)
  - Submission: students filling google forms for each paper (due the same week)

Invited Presentations

- Oct 27: Pre-training & Prompting (Section 2)
  - Lecture by Dr. Pengfei Liu (http://pfliu.com/), postdoc at CMU LTI
  - http://pretrain.nlpedia.ai/
- TBD: “No Language Left Behind” by Meta/Facebook AI (Section 1)
- TBD: An industrial talk from Deltek

CS678 Advanced Natural Language Processing

Structured Prediction 1: Sequence Tagging & Syntactic Parsing

Antonis Anastasopoulos & Ziyu Yao

https://nlp.cs.gmu.edu/course/cs678-fall22

Acknowledgement: Many slides are taken from Greg Durrett CS388@UT Austin and Antonios Anastasopoulos CS695@GMU

Structured Prediction

- Sequence labeling
  - POS Tagging & Hidden Markov Models (HMMs)
  - Named Entity Recognition & Conditional Random Fields (CRFs)
- Syntactic Parsing
  - Dependency parsing
  - Constituency parsing (next lecture)
- Semantic Parsing (next lecture)

Part-of-Speech Tagging

Linguistic Structures

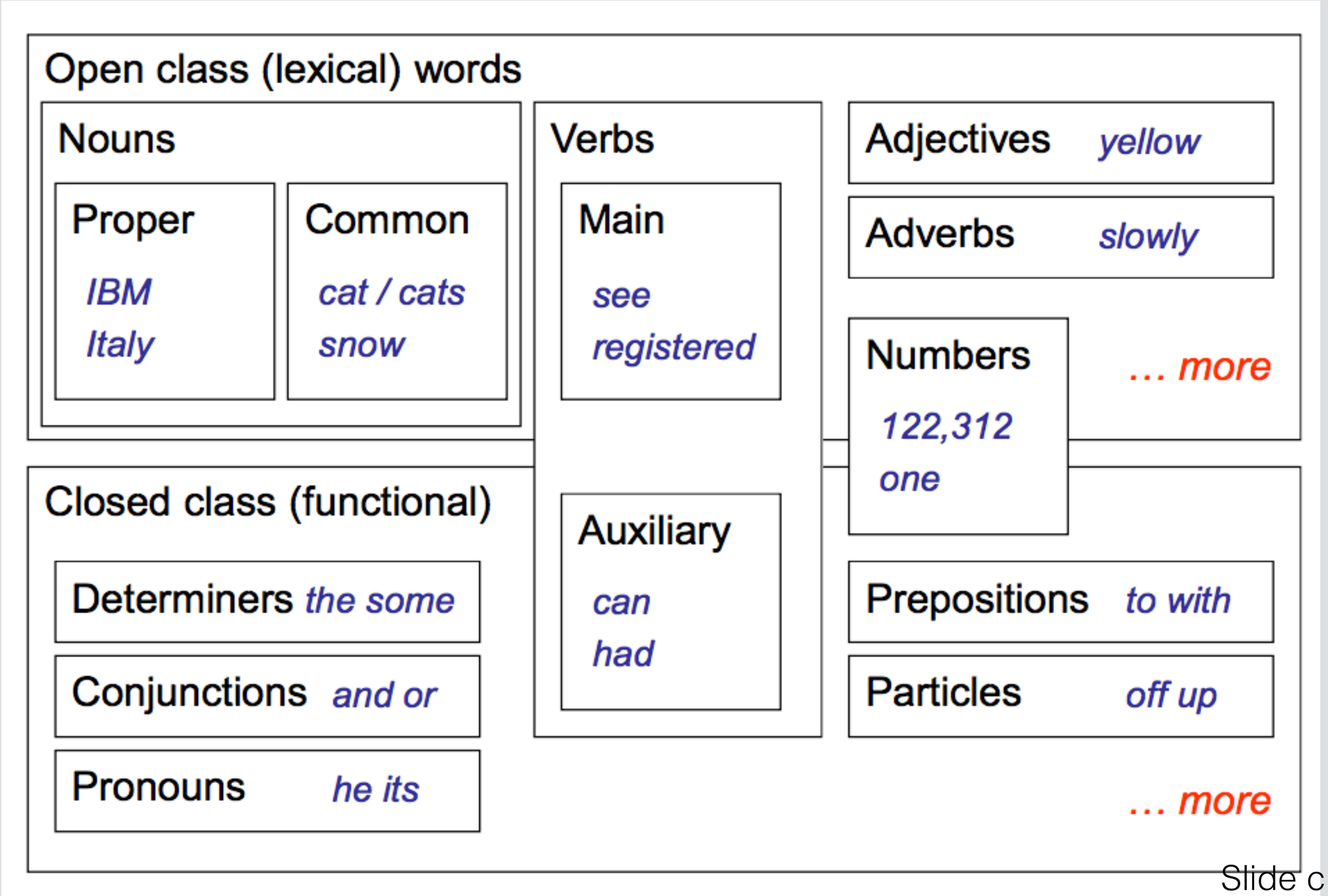
- Language has hierarchical structure, can represent with trees

- Understanding syntax fundamentally requires trees — the sentences have the same shallow analysis. But the first step we'll take towards understanding this is understanding parts of speech

NN NNS VBZ NNS

Teacher strikes idle kids

POS Tagging



Slide credit: Dan Klein

POS Tagging

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Figure 8.2 Penn Treebank part-of-speech tags.

POS Tagging

Ghana 's ambassador should have set up the big meeting in DC yesterday .

NNP POS NN MD VB VBN RP DT JJ NN IN NNP NN .

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

Figure 8.2 Penn Treebank part-of-speech tags.

POS Tagging

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN
Fed raises interest rates 0.5 percent

I hereby increase interest rates 0.5%

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN
Fed raises interest rates 0.5 percent

I'm 0.5% interested in the Fed's raises!

- Other paths are also plausible but even more semantically weird…
- What governs the correct choice? **Word + context**
  - Word identity: most words have <=2 tags, many have one (percent, the)
  - Context: nouns start sentences, nouns follow verbs, etc.

What is this good for?

- Text-to-speech: record, lead
- Preprocessing step for syntactic parsers or other tasks
- (Very) shallow information extraction

Hidden Markov Models

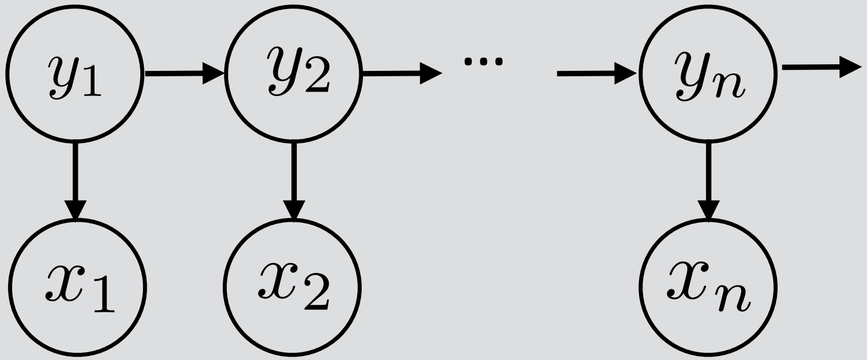
- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- Model the sequence of tags **y** over words **x** as a Markov process
- Markov property: future is conditionally independent of the past given the present

$$P(y_3|y_1, y_2) = P(y_3|y_2)$$

- If **y** are tags, this roughly corresponds to assuming that the next tag only depends on the current tag, not anything before

Hidden Markov Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$ $y_i \in$ T = set of possible tags (including STOP); $x_i \in$ V = vocab of words



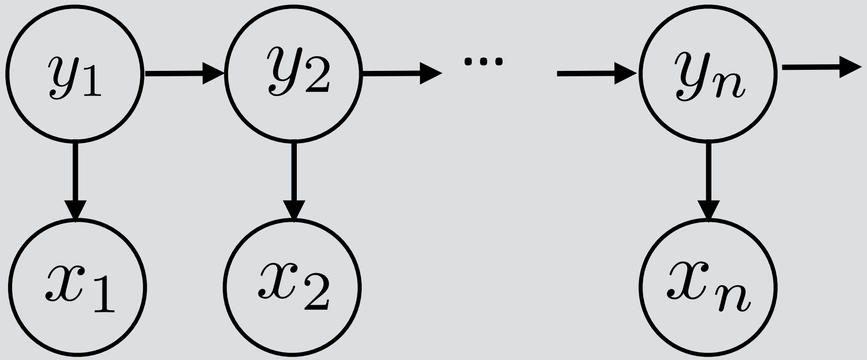
$$P(\mathbf{y}, \mathbf{x}) = \underbrace{P(y_1)}_{\text{Initial distribution}} \underbrace{\prod_{i=2}^{n} P(y_i|y_{i-1})}_{\text{Transition probabilities}} \underbrace{\prod_{i=1}^{n} P(x_i|y_i)}_{\text{Emission probabilities}}$$

Observation (x_i) depends only on current state (y_i)

HMM: Parameters

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$

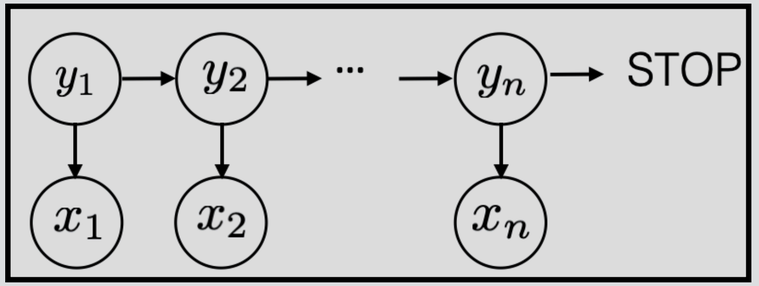
$y_i \in$ T = set of possible tags (including STOP);
$x_i \in$ V = vocab of words



$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} P(y_i|y_{i-1}) \prod_{i=1}^{n} P(x_i|y_i)$$

- Initial distribution: |T| x 1 vector (distribution over initial states)
- Emission distribution: |T| x |V| matrix (distribution over words per tag)
- Transition distribution: |T| x |T| matrix (distribution over next tags per tag)

HMMs Example



Tags = {N, V, STOP}

Vocabulary = {they, can, fish}

Initial

y_1	
N	1.0
V	0
STOP	0

Transition

y_{i-1} \ y_i	N	V	STOP
N	1/5	3/5	1/5
V	1/5	1/5	3/5

Emission

y_i \ x_i	they	can	fish
N	1	0	0
V	0	1/2	1/2

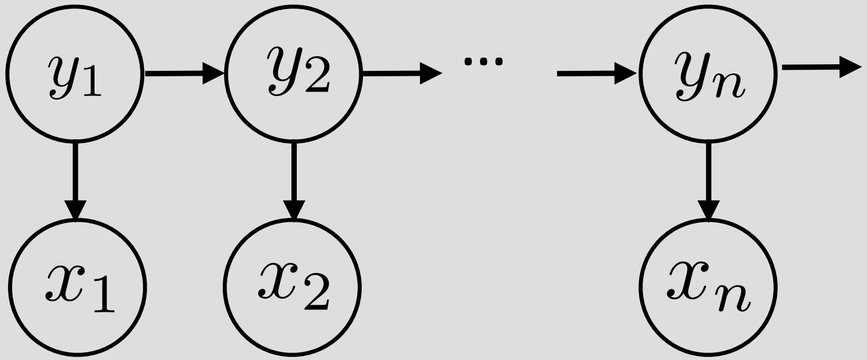
How are these probabilities learned?

Training HMMs

- Maximum Likelihood Estimation, essentially read counts + normalize
- Transitions: Count up all pairs (y_i, y_{i+1}) in the training data
  - Count up occurrences of what tag *T* can transition to
  - Normalize to get a distribution for P(*next tag* | *T*)
  - Need to *smooth* this distribution, won't discuss here
- Emissions: similar scheme, but trickier smoothing!
- Initials: even easier, just count up the currencies of possible first-position tags + doing smoothing

Inference in HMMs

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$



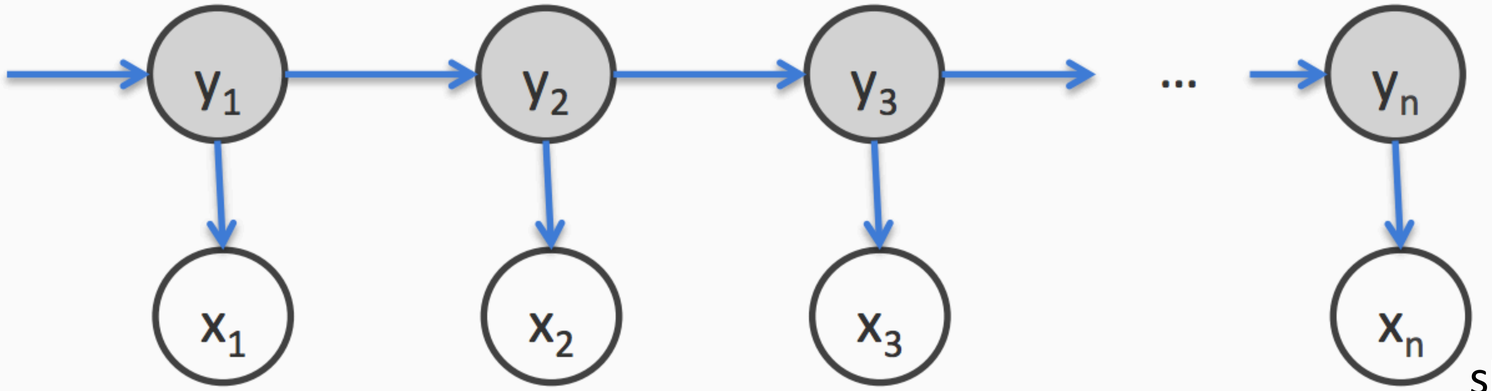
$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} P(y_i|y_{i-1}) \prod_{i=1}^{n} P(x_i|y_i)$$

- Inference problem: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Exponentially many possible ***y*** here!
- Solution: dynamic programming (possible because of Markov structure!)

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

Transition probabilities Emission probabilities Initial probability



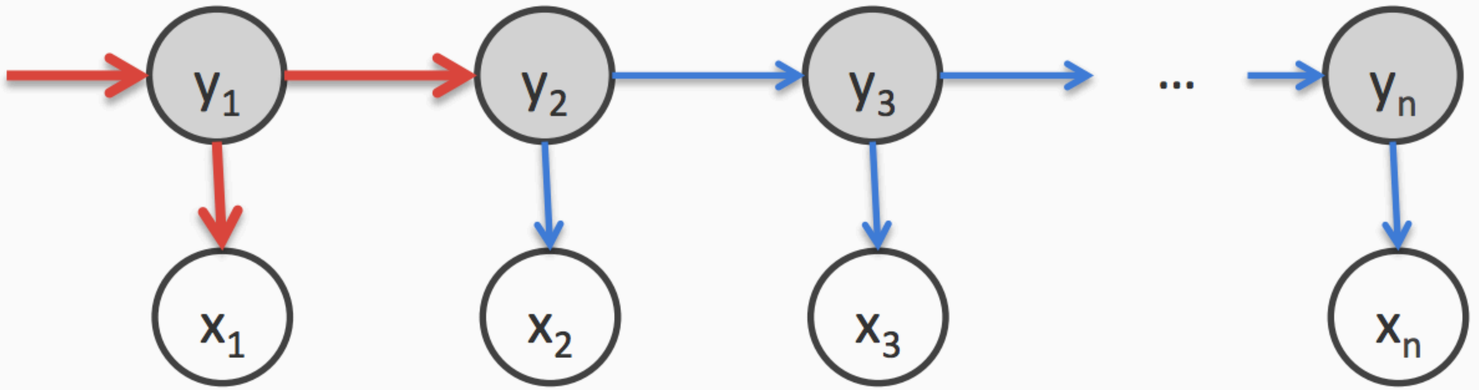
slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

The only terms that depend on y_1



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

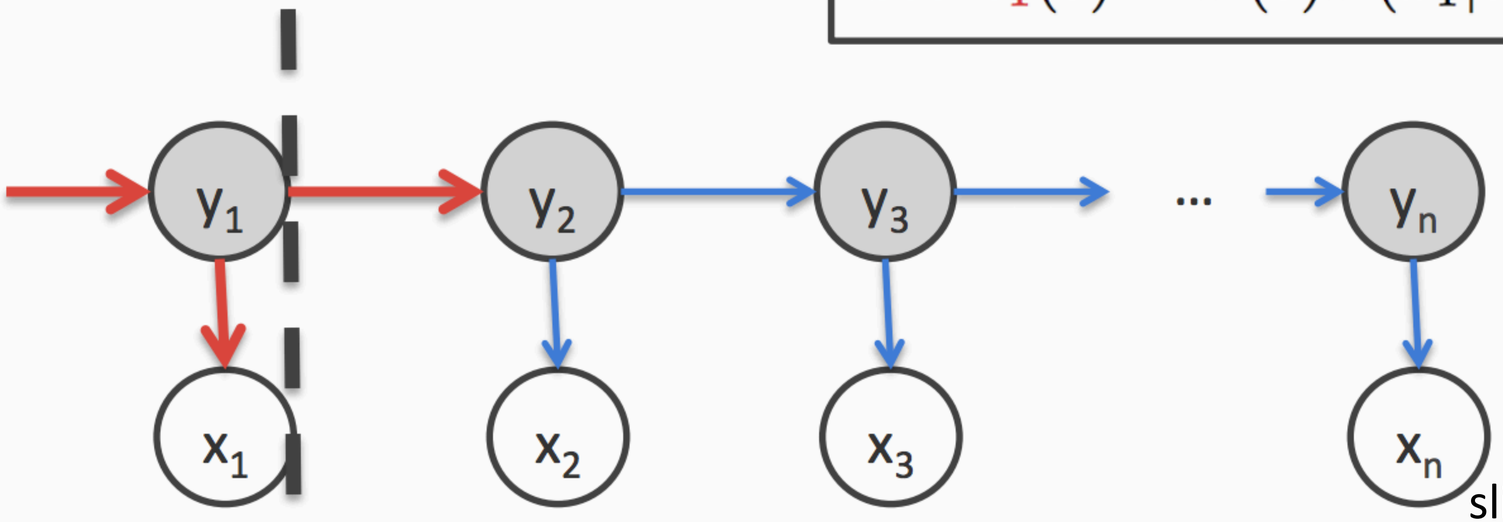
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

Abstract away the score for all decisions till here into score

- Best (partial) score for a sequence ending in state *s*

$$\text{score}_1(s) = P(s)P(x_1|s)$$



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

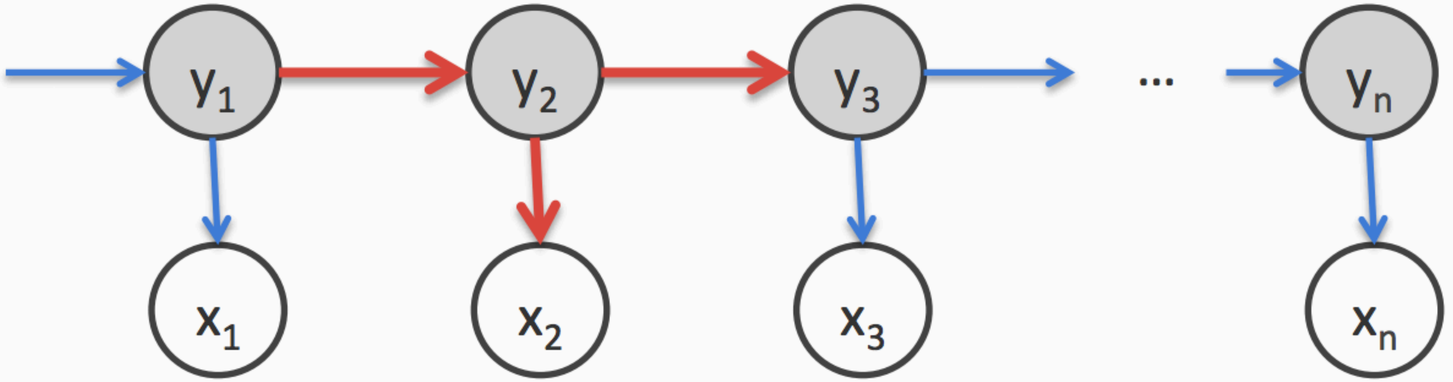
$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

Only terms that depend on y_2



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

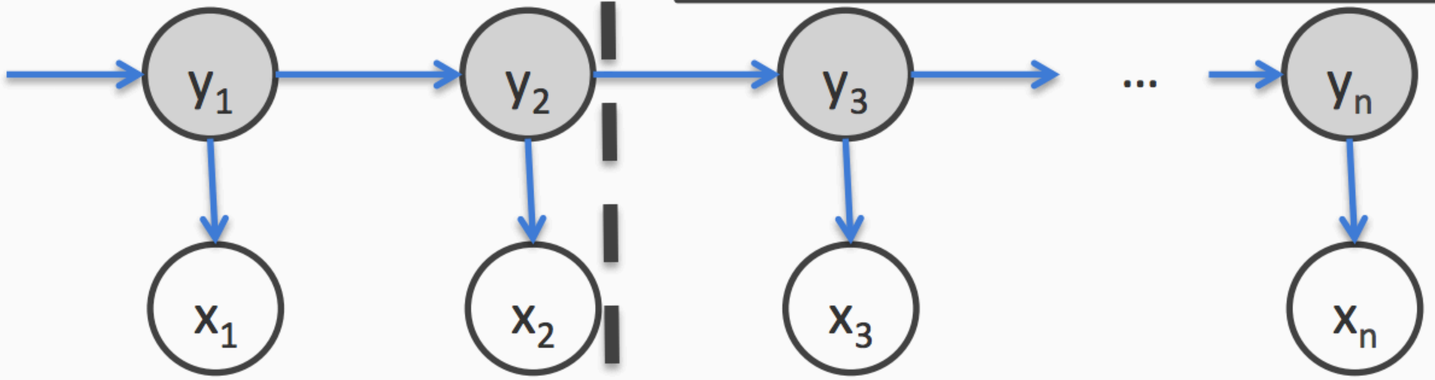
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\text{score}_2(y_2)$$

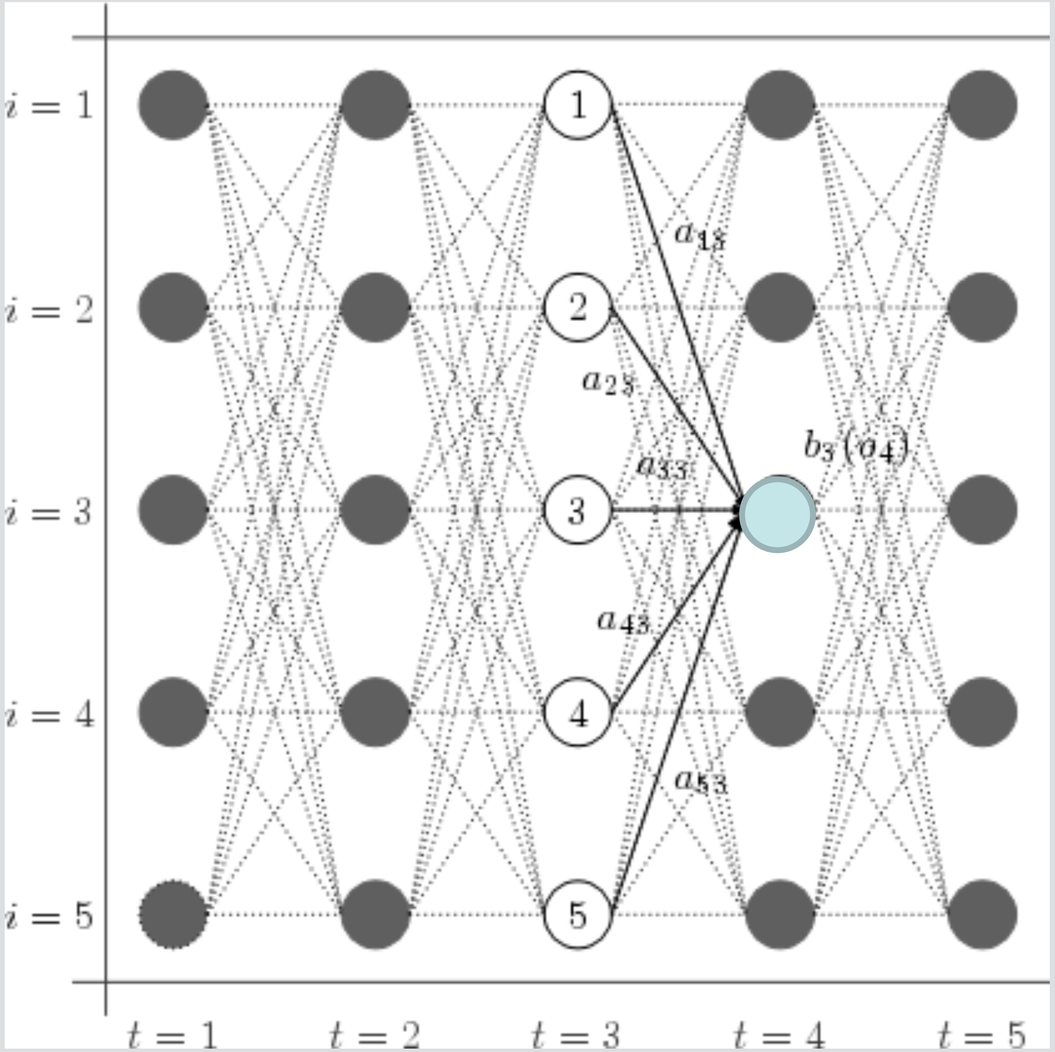
$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

Viterbi Algorithm



- “Think about” all possible immediate prior state values. Everything before that has already been accounted for by earlier stages.

slide credit: Dan Klein

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

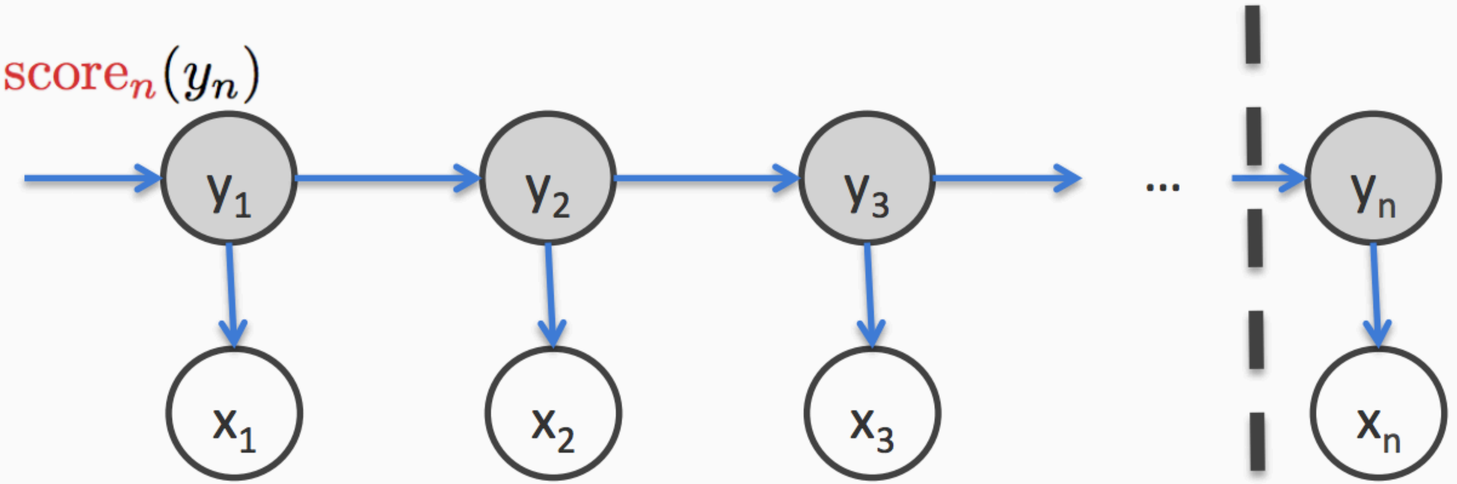
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \text{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \text{score}_n(y_n)$$



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\text{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \text{score}_n(y_n)$$

$$\text{score}_1(s) = P(s)P(x_1|s)$$

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

slide credit: Vivek Srikumar

1. Initial: For each state s, calculate

$$\text{score}_1(s) = P(s)P(x_1|s) = \pi_s B_{x_1,s}$$

2. Recurrence: For i = 2 to n, for every state s, calculate

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

$$= \max_{y_{i-1}} A_{y_{i-1},s} B_{s,x_i} \text{score}_{i-1}(y_{i-1})$$

3. Final state: calculate

$$\max_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}|\pi, A, B) = \max_{s} \text{score}_n(s)$$

π: Initial probabilities
A: Transitions
B: Emissions

This only calculates the max. To get final answer (*argmax*),

- keep track of which state corresponds to the max at each step
- build the answer using these back pointers

(JM Ch8.4.5 for details)

slide credit: Vivek Srikumar

HMM POS Tagging

- Baseline: assign each word its most frequent tag: ~90% accuracy
- HMM-based POS Taggers:
  - Trigram HMM: ~95% accuracy / 55% on unknown words
  - TnT tagger (Brants 1998, tuned HMM): 96.2% accuracy / 86.0% on unks
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein

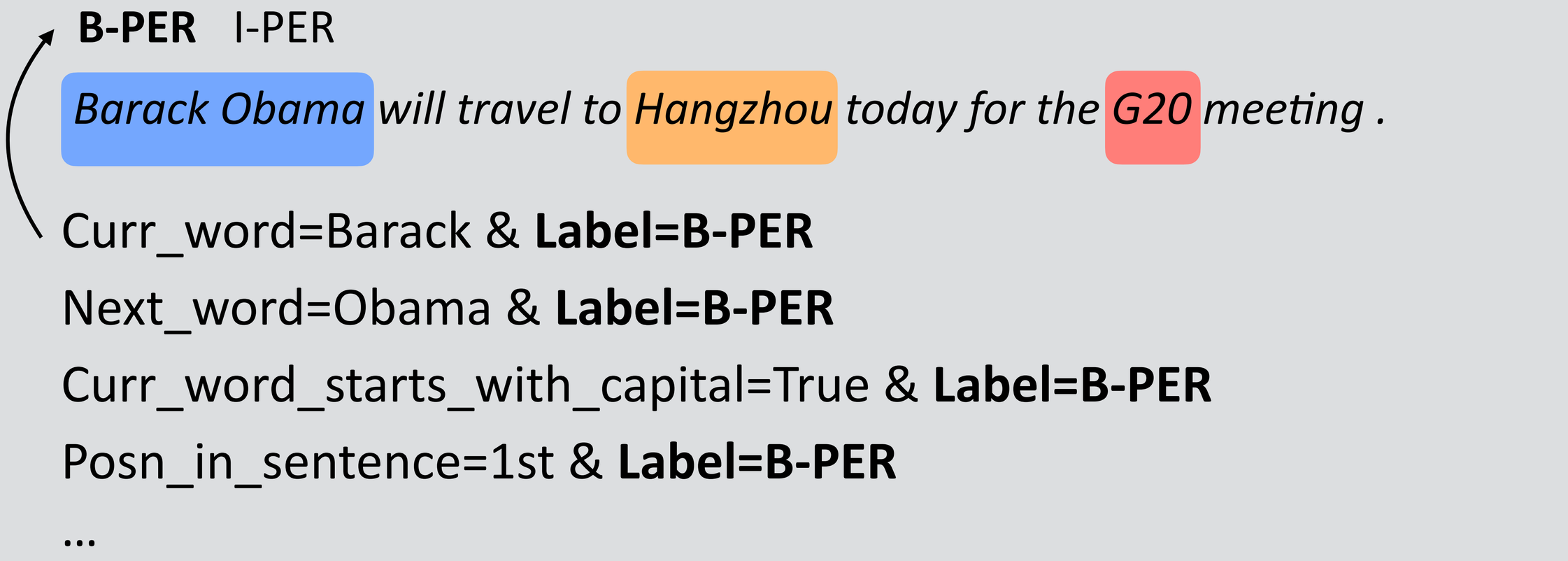
Named Entity Recognition

Named Entity Recognition

- BIO tag set: begin, inside, outside
- Sequence of tags — should we use an HMM?
- Why might an HMM not do so well here?
  - Lots of O's
  - Insufficient features/capacity with multinomials (especially for UNKs)

Conditional Random Fields

- Flexible *discriminative model* for tagging tasks that can use arbitrary *features* of the input. Similar to logistic regression, but structured



Tagging with Logistic Regression

- Logistic regression over each tag individually:

$$P(y_i = y|\mathbf{x}, i) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(y, i, \mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \mathbf{f}(y', i, \mathbf{x}))}$$

Probability of the *i*-th word getting assigned tag *y* (e.g., B-PER)

Tagging with Logistic Regression

- Logistic regression over each tag individually:

$$P(y_i = y|\mathbf{x}, i) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(y, i, \mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \mathbf{f}(y', i, \mathbf{x}))}$$

- Over all tags:

$$P(\mathbf{y} = \tilde{\mathbf{y}}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i = \tilde{y}_i|\mathbf{x}, i) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(\tilde{y}_i, i, \mathbf{x})\right)$$

Look like HMM's "emission" modeling

- Set Z equal to the product of denominators

- Score of a prediction: sum of weights dot features over each individual predicted tag (this is a simple CRF but not the general form)

Example: “Emission Features” f_e

B-PER I-PER O O

Barack Obama will travel

feats = $\mathbf{f}_e$(B-PER, i=1, **x**) + $\mathbf{f}_e$(I-PER, i=2, **x**) + $\mathbf{f}_e$(O, i=3, **x**) + $\mathbf{f}_e$(O, i=4, **x**)

[CurrWord=*Obama* & label=I-PER, PrevWord=*Barack* & label=I-PER, CurrWordIsCapitalized & label=I-PER, …]

B-PER B-PER O O

Barack Obama will travel

feats = $\mathbf{f}_e$(B-PER, i=1, **x**) + $\mathbf{f}_e$(B-PER, i=2, **x**) + $\mathbf{f}_e$(O, i=3, **x**) + $\mathbf{f}_e$(O, i=4, **x**)

Adding Structure

$$P(\mathbf{y} = \tilde{\mathbf{y}}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(\tilde{y}_i, i, \mathbf{x})\right)$$

- We want to be able to learn that some tags don't follow other tags — want to have features on *tag pairs*

$$P(\mathbf{y} = \tilde{\mathbf{y}}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}_e(\tilde{y}_i, i, \mathbf{x}) + \sum_{i=2}^{n} \mathbf{w}^\top \mathbf{f}_t(\tilde{y}_{i-1}, \tilde{y}_i, i, \mathbf{x})\right)$$

- Score: sum of weights dot **f_e** features over each predicted tag ("emissions") plus sum of weights dot **f_t** features over tag pairs ("transitions")
- This is a *sequential CRF*

Example

B-PER I-PER O O

Barack Obama will travel

feats = $\mathbf{f}_e$(B-PER, i=1, **x**) + $\mathbf{f}_e$(I-PER, i=2, **x**) + $\mathbf{f}_e$(O, i=3, **x**) + $\mathbf{f}_e$(O, i=4, **x**)
+ $\mathbf{f}_t$(B-PER, I-PER, i=1, **x**) + $\mathbf{f}_t$(I-PER, O, i=2, **x**) + $\mathbf{f}_t$(O, O, i=3, **x**)

B-PER B-PER O O

Barack Obama will travel

feats = $\mathbf{f}_e$(B-PER, i=1, **x**) + $\mathbf{f}_e$(B-PER, i=2, **x**) + $\mathbf{f}_e$(O, i=3, **x**) + $\mathbf{f}_e$(O, i=4, **x**)
+ $\mathbf{f}_t$(B-PER, B-PER, i=1, **x**) + $\mathbf{f}_t$(B-PER, O, i=2, **x**) + $\mathbf{f}_t$(O, O, i=3, **x**)

- *Obama* can start a new named entity (emission feats look okay), but we're not likely to have two PER entities in a row (transition feats)

CRFs Outline

- Model:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x}) \right]$$

(Linear feature)

- Inference
- Learning

Inference

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

- $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$: can use Viterbi exactly as in HMM case

$$\max_{y_1,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} e^{\phi_t(y_1,y_2)} e^{\phi_e(y_1,1,\mathbf{x})}$$

- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, use the exact same Viterbi dynamic program

CRFs Outline

- Model:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x}) \right]$$

- Inference: argmax P(**y**|**x**) from Viterbi

- Learning

Learning

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x}) \right]$$

- Given a training set of (**x**, **y***) pairs, maximize $\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \log P(\mathbf{y}^*|\mathbf{x})$
- Gradient is analogous to logistic regression: gold feats - expected feats

$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=2}^{n} f_t(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} f_e(y^*_i, i, \mathbf{x})$$

$$-\mathbb{E}_{\mathbf{y}} \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x}) \right]$$

intractable!

Training CRFs

$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=2}^{n} f_t(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} f_e(y^*_i, i, \mathbf{x})$$
$$-\mathbb{E}_{\mathbf{y}}\left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right]$$

- Let's consider the emission feature expectation first

$$\mathbb{E}_{\mathbf{y}}\left[\sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right] = \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x})\left[\sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right] = \sum_{i=1}^{n}\sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x}) f_e(y_i, i, \mathbf{x})$$

$$= \sum_{i=1}^{n}\sum_{s} \boxed{P(y_i = s|\mathbf{x})} f_e(s, i, \mathbf{x})$$

How to calculate this marginal probability?

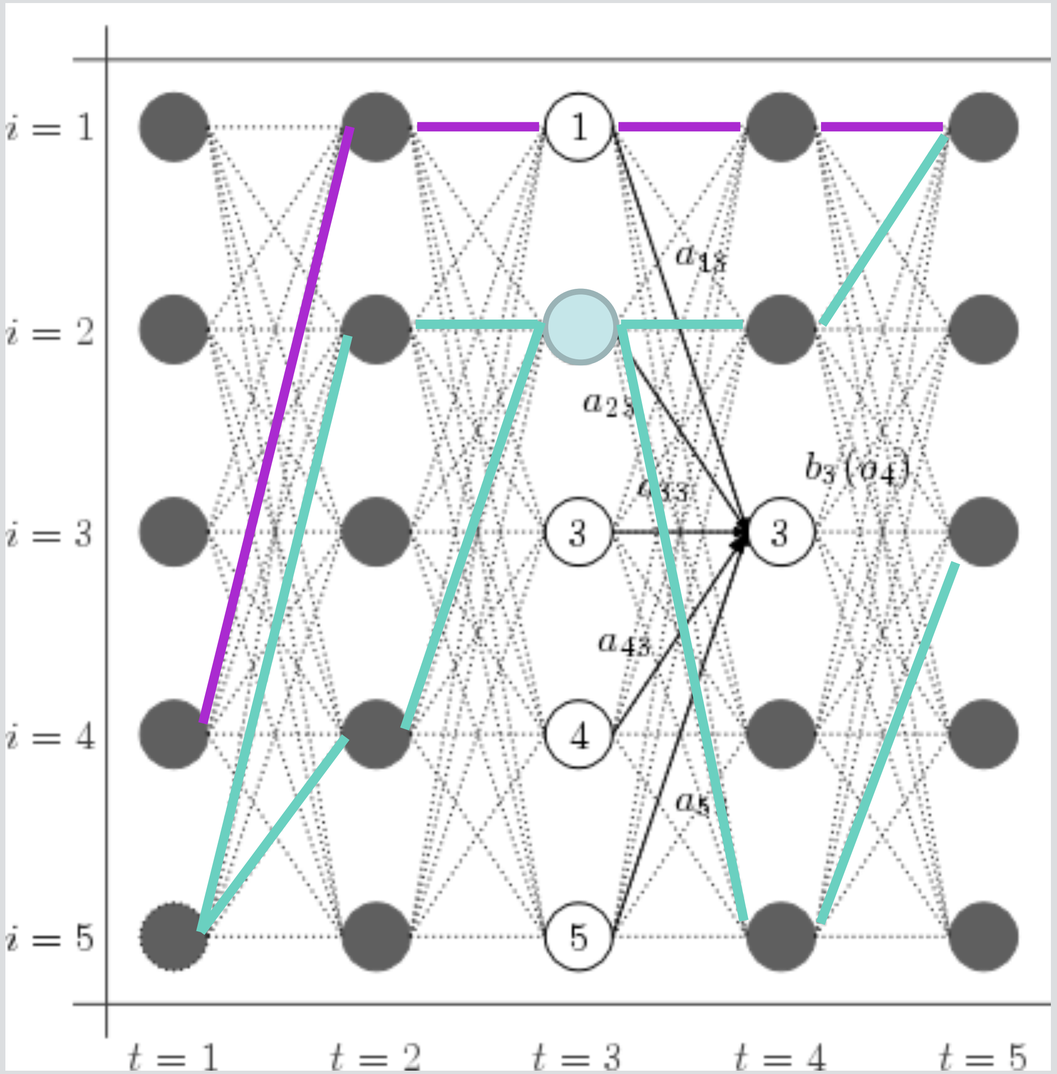
Forward-Backward Algorithm

- How do we compute these marginals $P(y_i = s|\mathbf{x})$?

$$P(y_i = s|\mathbf{x}) = \sum_{y_1,\cdots,y_{i-1},y_{i+1},\cdots,y_n} P(\mathbf{y}|\mathbf{x})$$

- What did Viterbi compute? $P(\mathbf{y}_{\max}|\mathbf{x}) = \max_{y_1,\cdots,y_n} P(\mathbf{y}|\mathbf{x})$
- Can compute marginals with dynamic programming as well using the forward-backward algorithm

Forward-Backward Algorithm

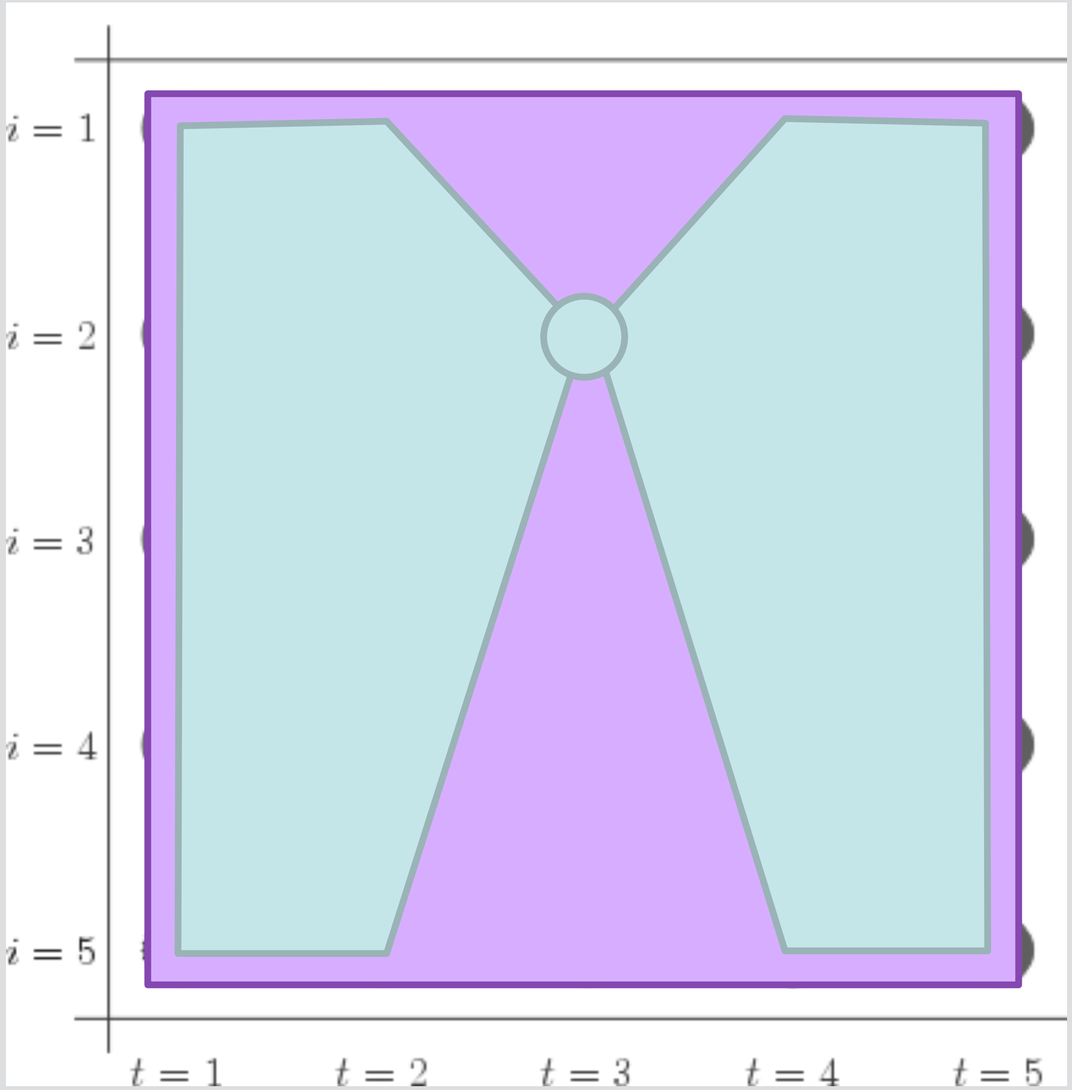


"2" is like a candidate NER tag (e.g., B-PER)

$$P(y_3 = 2|\mathbf{x}) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

slide credit: Dan Klein

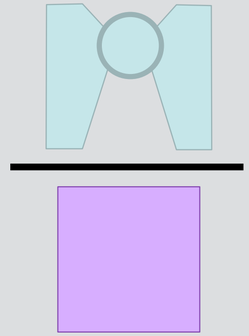
Forward-Backward Algorithm



$$P(y_3 = 2|\mathbf{x}) =$$

$$\frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

=



- Easiest and most flexible to do one pass to compute  and one to compute 

(Eisenstein Ch7.5.3 for details)

slide credit: Dan Klein

Training CRFs

- For emission features:

$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^{n}\sum_{s} P(y_i = s|\mathbf{x}) f_e(s, i, \mathbf{x})$$

  - Need compute $P(y_i = s|\mathbf{x})$

- For transition features: similarly compute $P(y_i = s_1, y_{i+1} = s_2|\mathbf{x})$ using forward-backward algorithm

- …but you can build a pretty good system without learned transition features (use heuristic weights, or just enforce constraints like B-PER -> I-ORG is illegal)

CRFs Outline

- Model:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right]$$

- Inference: argmax P(**y**|**x**) from Viterbi

- Learning: run forward-backward to compute the marginals

$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^{n}\sum_{s} P(y_i = s|\mathbf{x}) f_e(s, i, \mathbf{x})$$

Learning: Pseudocode

- for each epoch
  - for each example
    - extract features on each emission and transition (look up in cache)
    - compute marginal probabilities with forward-backward
    - compute phis (ϕ_t, ϕ_e) based on features and weights
    - accumulate gradient over all emissions and transitions
    - Perform gradient ascent

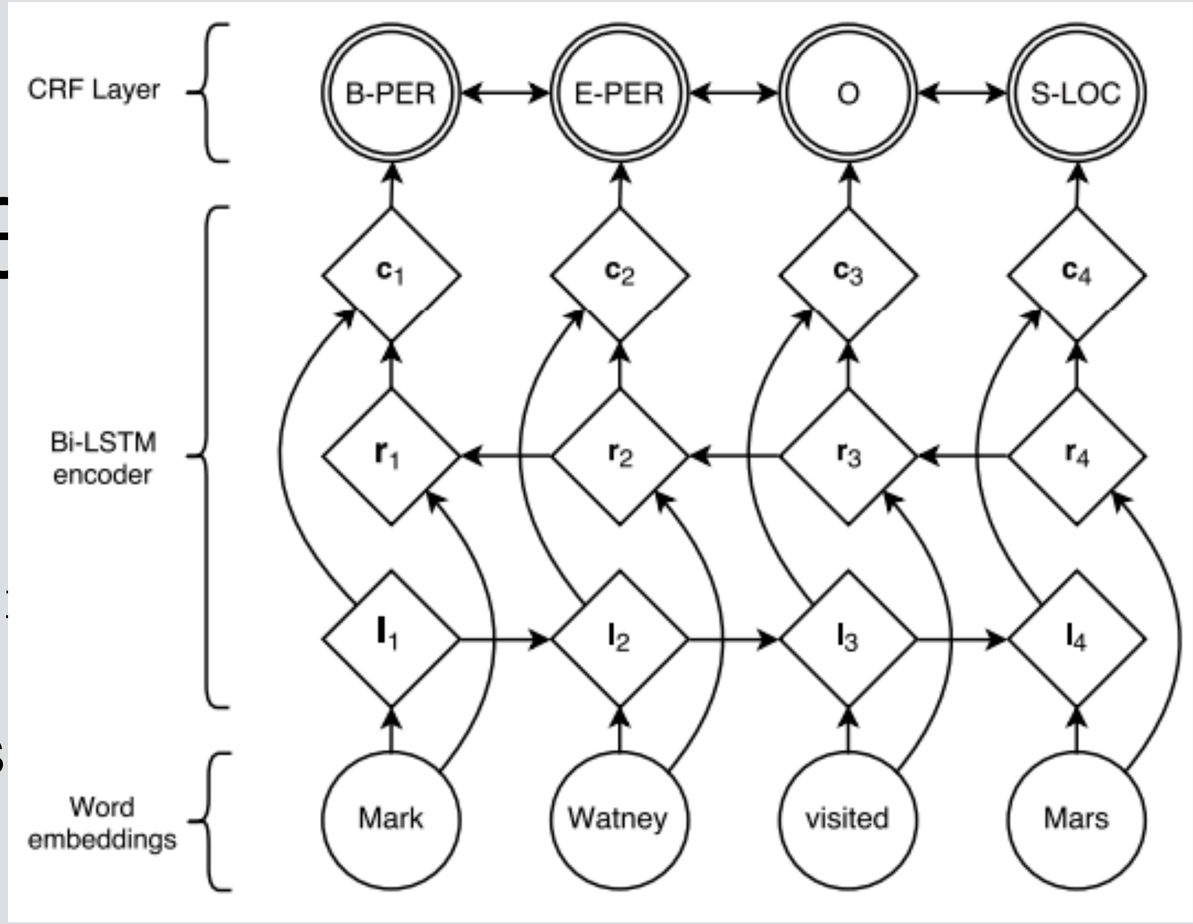
$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^{n}\sum_{s} P(y_i = s|\mathbf{x}) f_e(s, i, \mathbf{x})$$

Last Note: Ne

- So far, linear features

$$P(\mathbf{y} = \tilde{\mathbf{y}}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}_e(\tilde{y}_i, i, \right.$$

- How about nonlinear features, e.g., us
  neural nets?
  - e.g., can we use RNN or LSTM to learn neural features for CRF?
  - Lample et al., 2016 (F1 vs. SOTA using hand-crafted features): 90.94 (en, vs. 91.2), 78.76 (de, vs. 76.22), 81.74 (nl, vs. 82.84), 85.75 (es, vs. 82.95)



Neural Architectures for Named Entity Recognition, Lample et al., 2016

Structured Prediction

- Sequence labeling
  - POS Tagging & Hidden Markov Models (HMMs)
  - Named Entity Recognition & Conditional Random Fields (CRFs)
- Syntactic Parsing
  - Dependency parsing
  - Constituency parsing (next lecture)
- Semantic Parsing (next lecture)

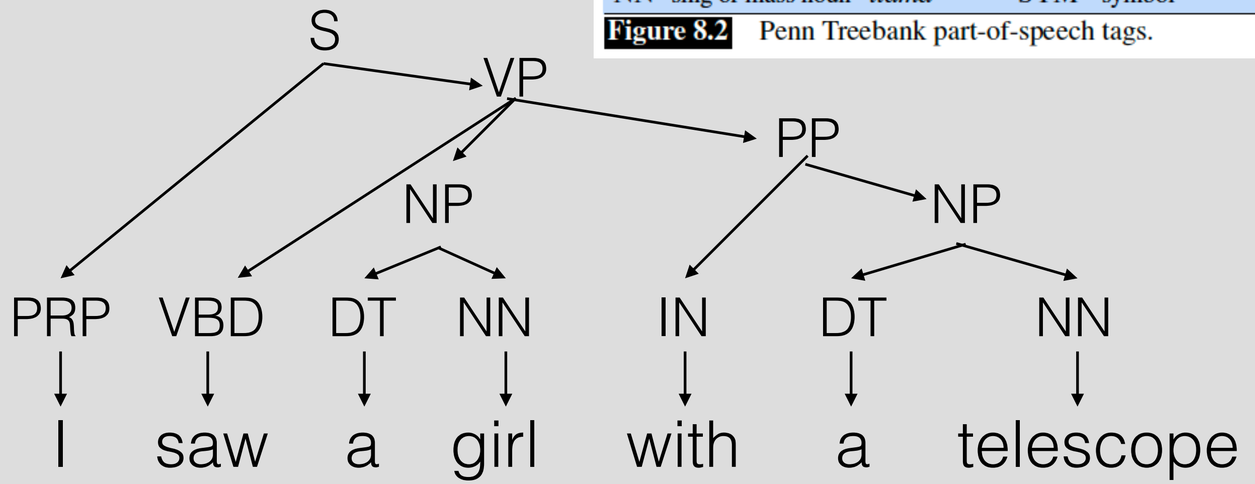
Syntactic Parsing

Syntax

- Study of word order and how words form sentences
- Why do we care about syntax?
  - Recognize verb-argument structures (who is doing what to whom?)
  - Multiple interpretations of words (noun or verb?)
  - Higher level of abstraction beyond words: some languages are SVO, some are VSO, some are SOV, parsing can canonicalize

Two Ty
Linguistic

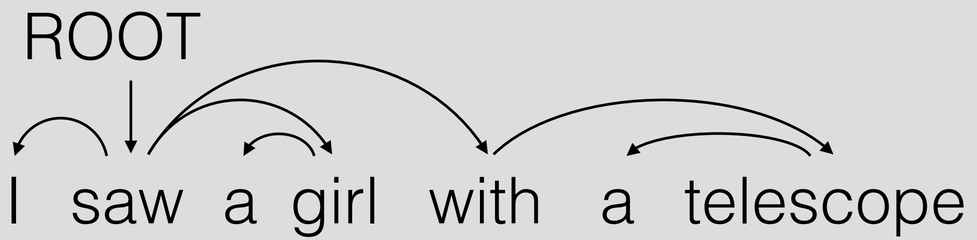
- **Phrase structure**: focus on the stru



Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past participle	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	+,%, &	WRB	wh-adverb	*how, where*

Figure 8.2 Penn Treebank part-of-speech tags.

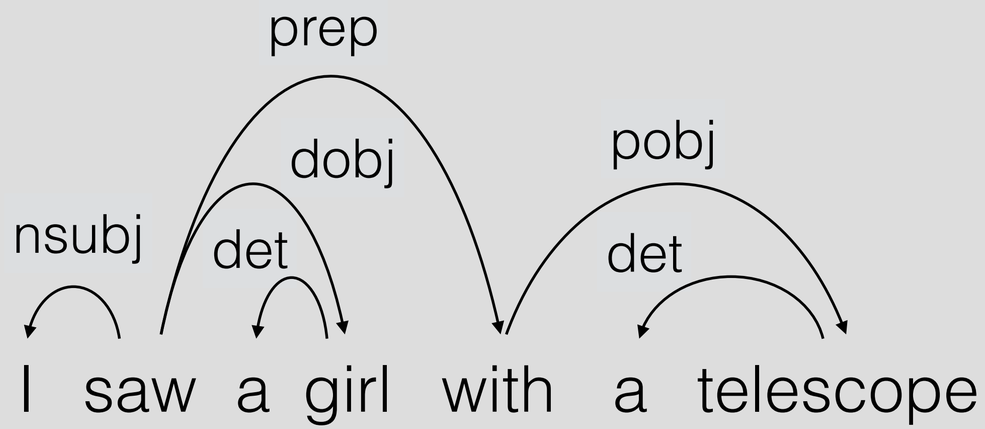
- **Dependency:** focus on relations between words



Syntactic Parsing 1: Dependency Parsing

Why Dependencies?

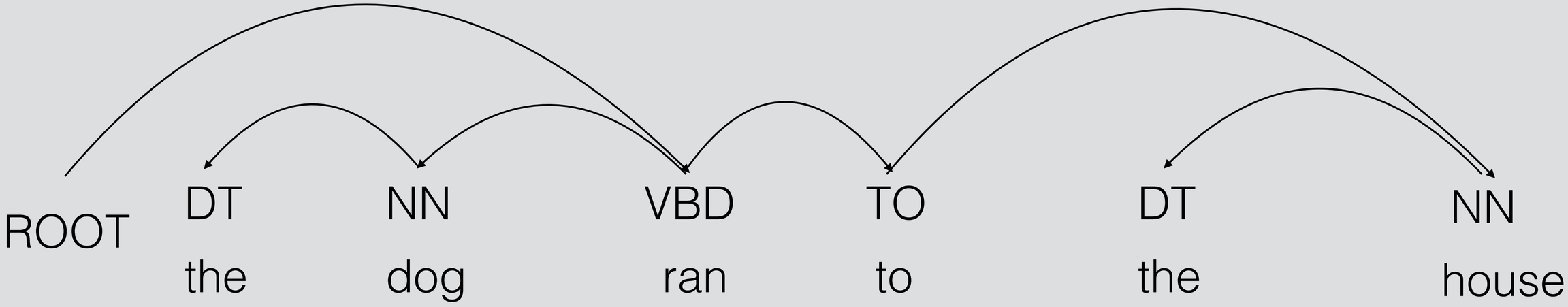
- Demonstrate the relationships between words in a straightforward way



- Particularly good for multilinguality, e.g. phrase-structure can be hard to define in languages with free word order

Dependency Parsing

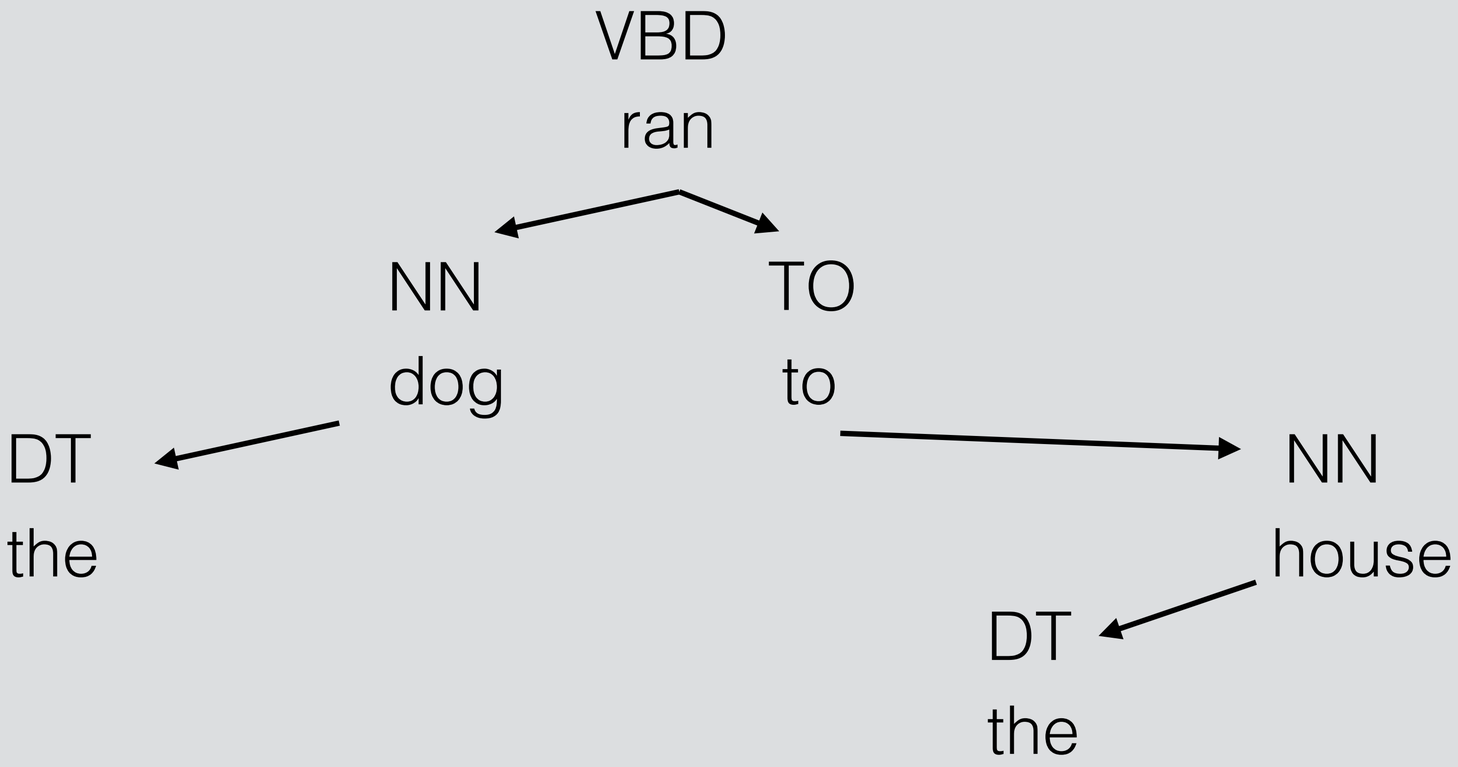
- Dependency syntax: syntactic structure is defined by these arcs
  - *Head (parent, governor)* connected to *dependent (child, modifier)*
  - Each word has exactly one parent except for the *ROOT* symbol, dependencies must form a directed acyclic graph



- POS tags same as before, usually run a tagger first as preprocessing

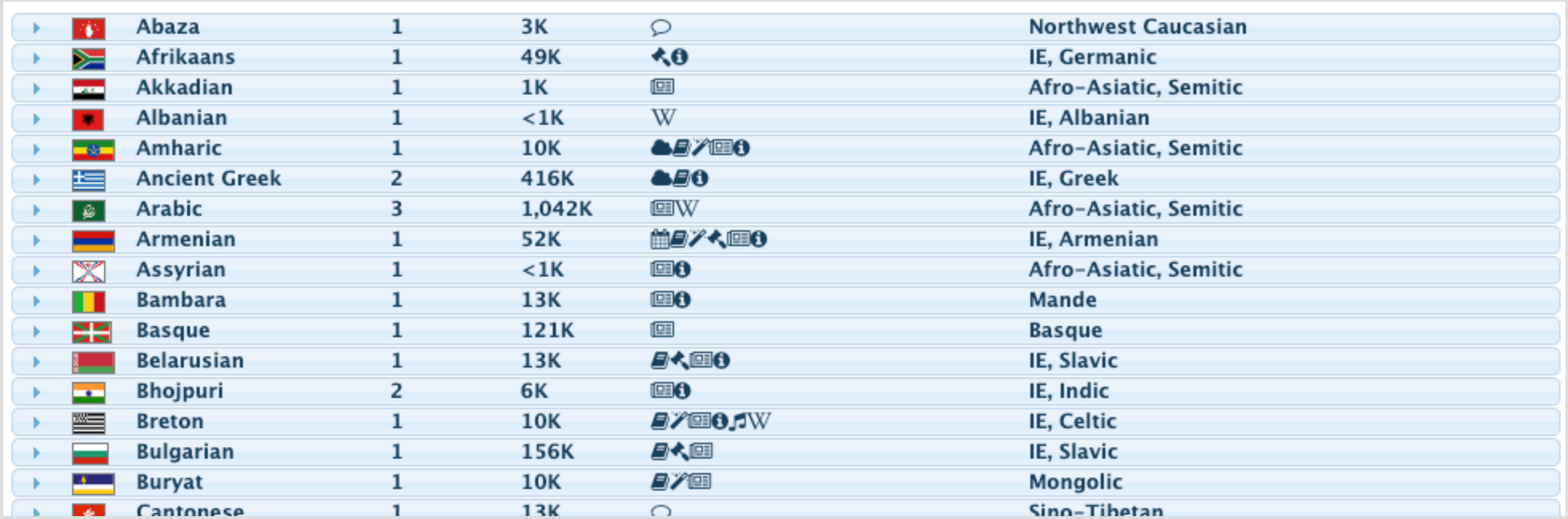
Dependency Parsing

- Still a notion of hierarchy! Subtrees often align with constituents



Universal Dependencies Treebank

- Standard format for parse trees in many languages

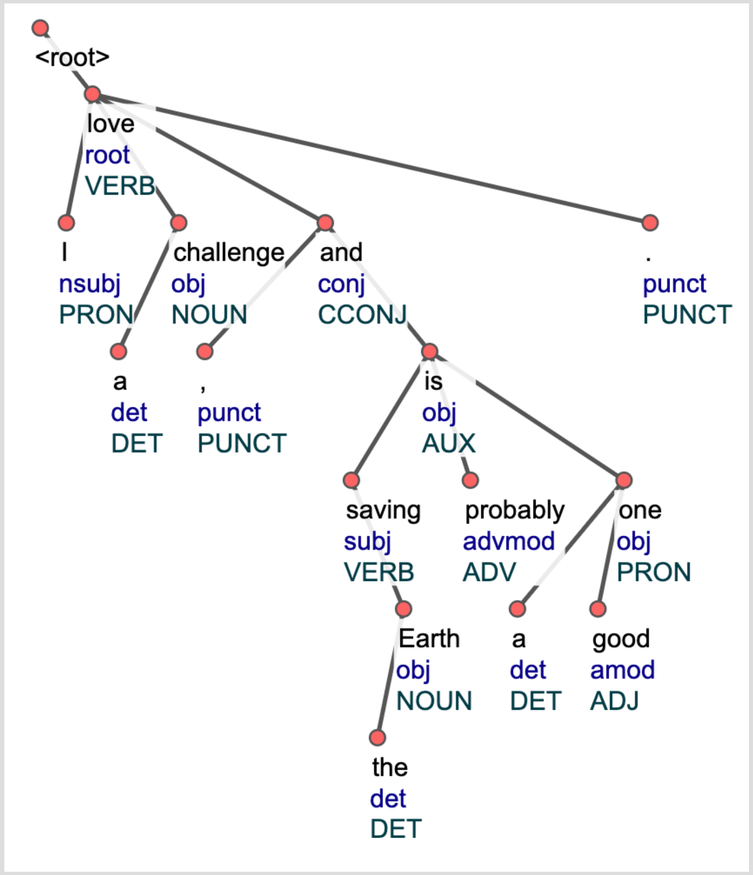


Abaza	1	3K	Northwest Caucasian
Afrikaans	1	49K	IE, Germanic
Akkadian	1	1K	Afro-Asiatic, Semitic
Albanian	1	<1K	IE, Albanian
Amharic	1	10K	Afro-Asiatic, Semitic
Ancient Greek	2	416K	IE, Greek
Arabic	3	1,042K	Afro-Asiatic, Semitic
Armenian	1	52K	IE, Armenian
Assyrian	1	<1K	Afro-Asiatic, Semitic
Bambara	1	13K	Mande
Basque	1	121K	Basque
Belarusian	1	13K	IE, Slavic
Bhojpuri	2	6K	IE, Indic
Breton	1	10K	IE, Celtic
Bulgarian	1	156K	IE, Slavic
Buryat	1	10K	Mongolic
Cantonese	1	13K	Sino-Tibetan

https://universaldependencies.org/

Semantic and Syntactic Dependencies

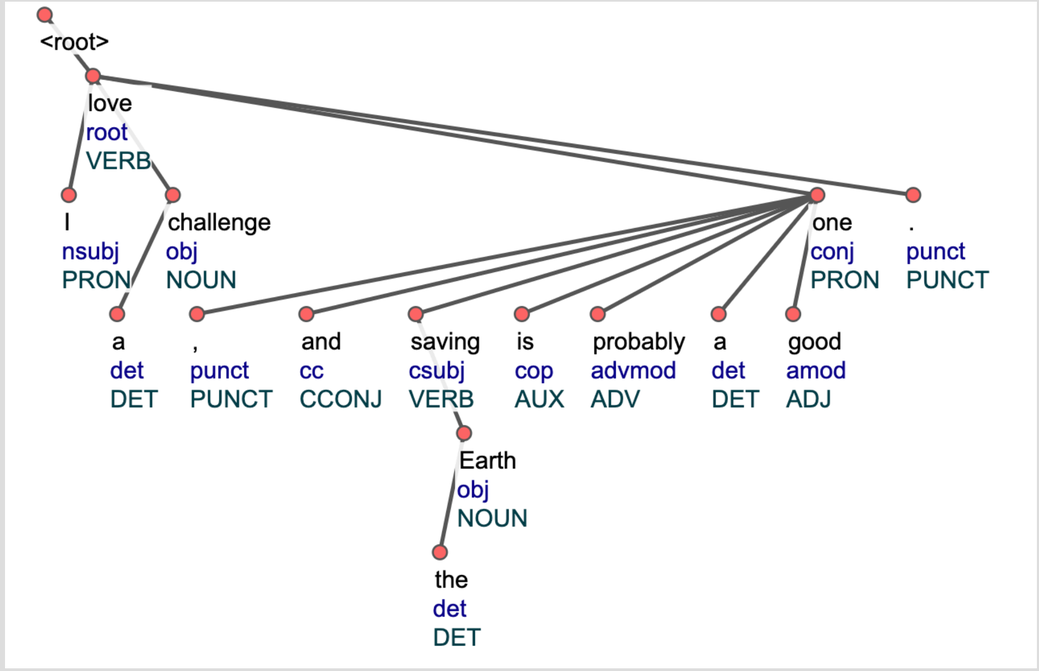
Syntactic (SUD)



Deeper, reflect phrase structure, more function word heads

https://surfacesyntacticud.github.io/

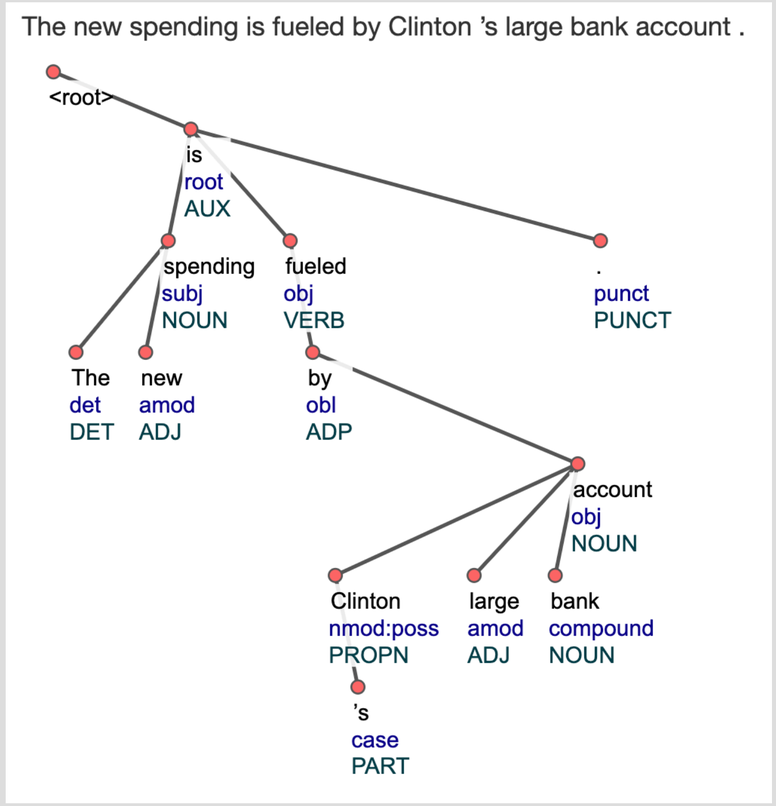
Semantic (UD)



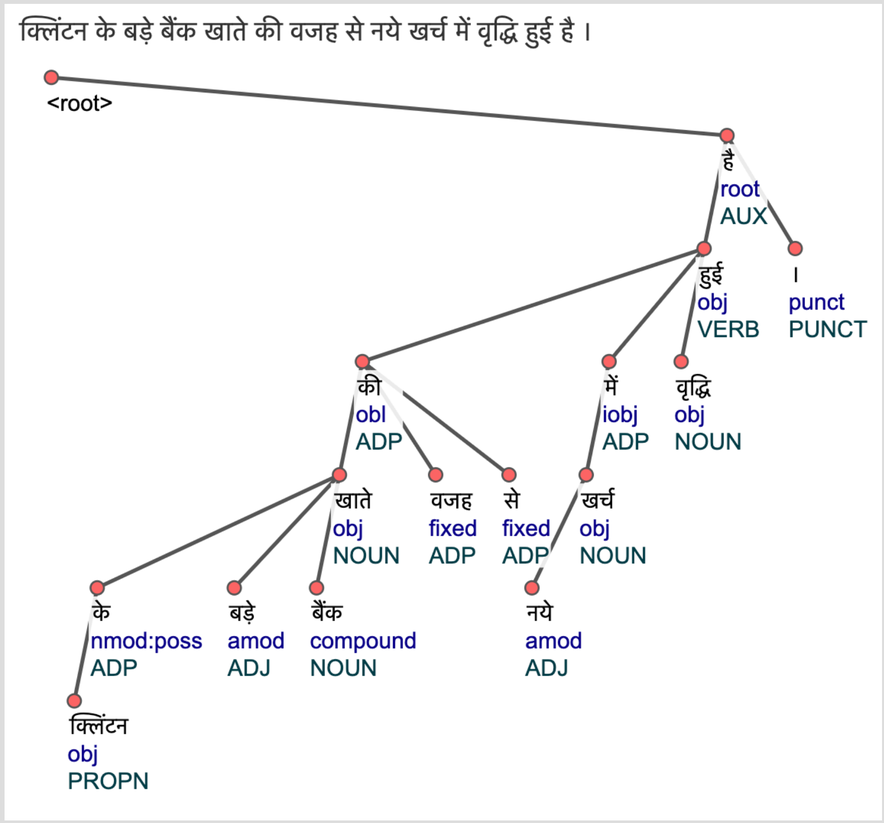
Flatter, semantically related words closer, more content word heads

Cross-lingual Differences In Structure

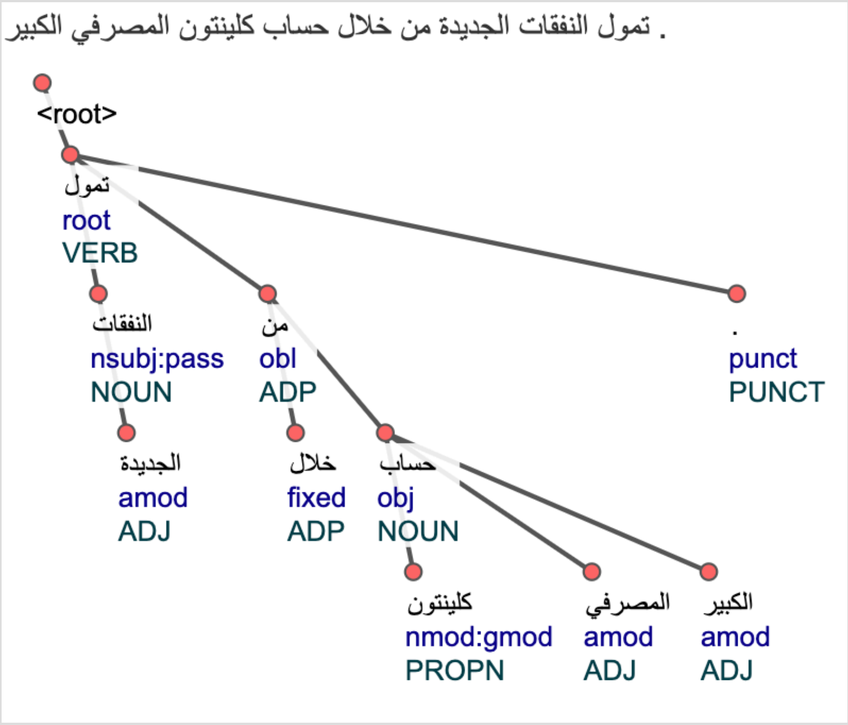
English: SVO



Hindi: Verb Final



Arabic: Verb Initial



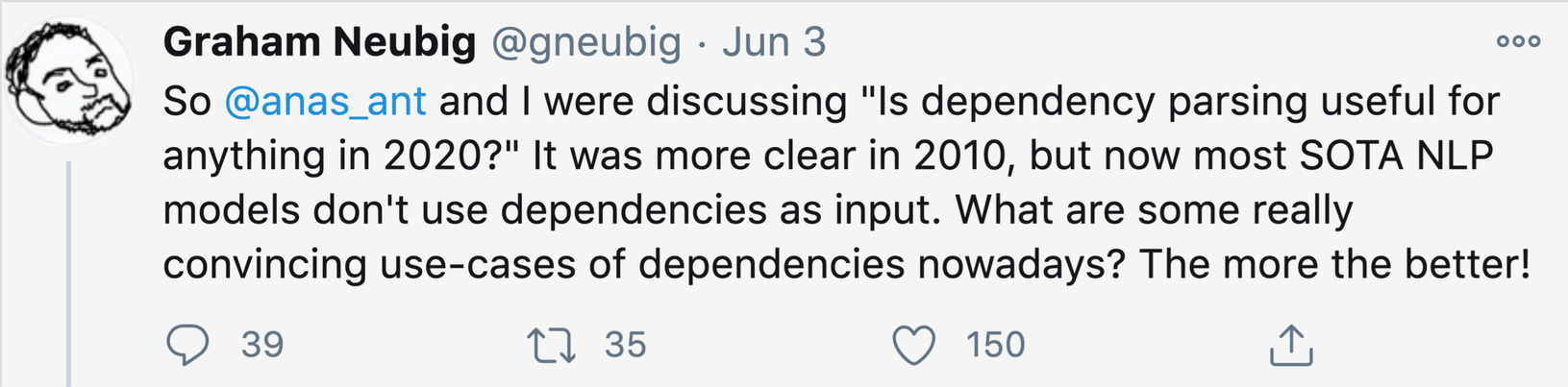
https://universaldependencies.org/treebanks/{en,hi,ar}_pud/index.html

What Can we Do w/ Dependencies?

Use Cases of Dependencies?

- Previously, used for feature engineering in systems (and still useful in some cases)

- Now: more useful for human-facing applications

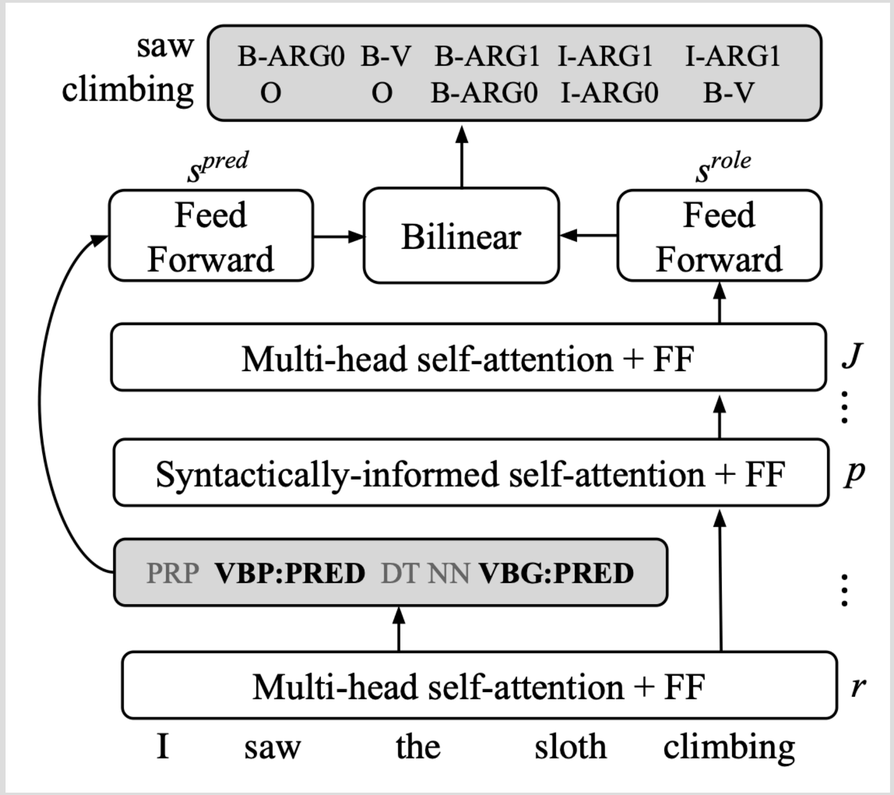


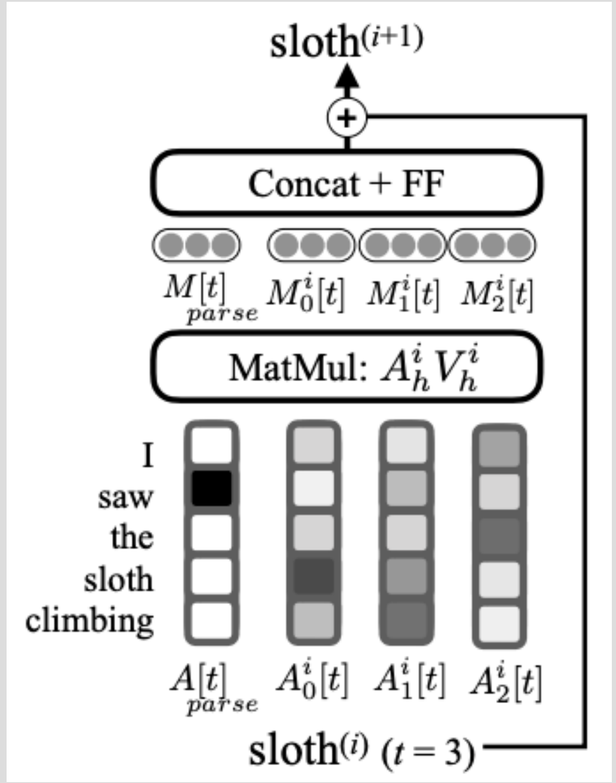
https://twitter.com/gneubig/status/1268238606101032962?lang=en

Slides credit: Antonios Anastasopoulos

Example 1: Adding Inductive Bias to Neural Models

- Bias self attention to follow syntax

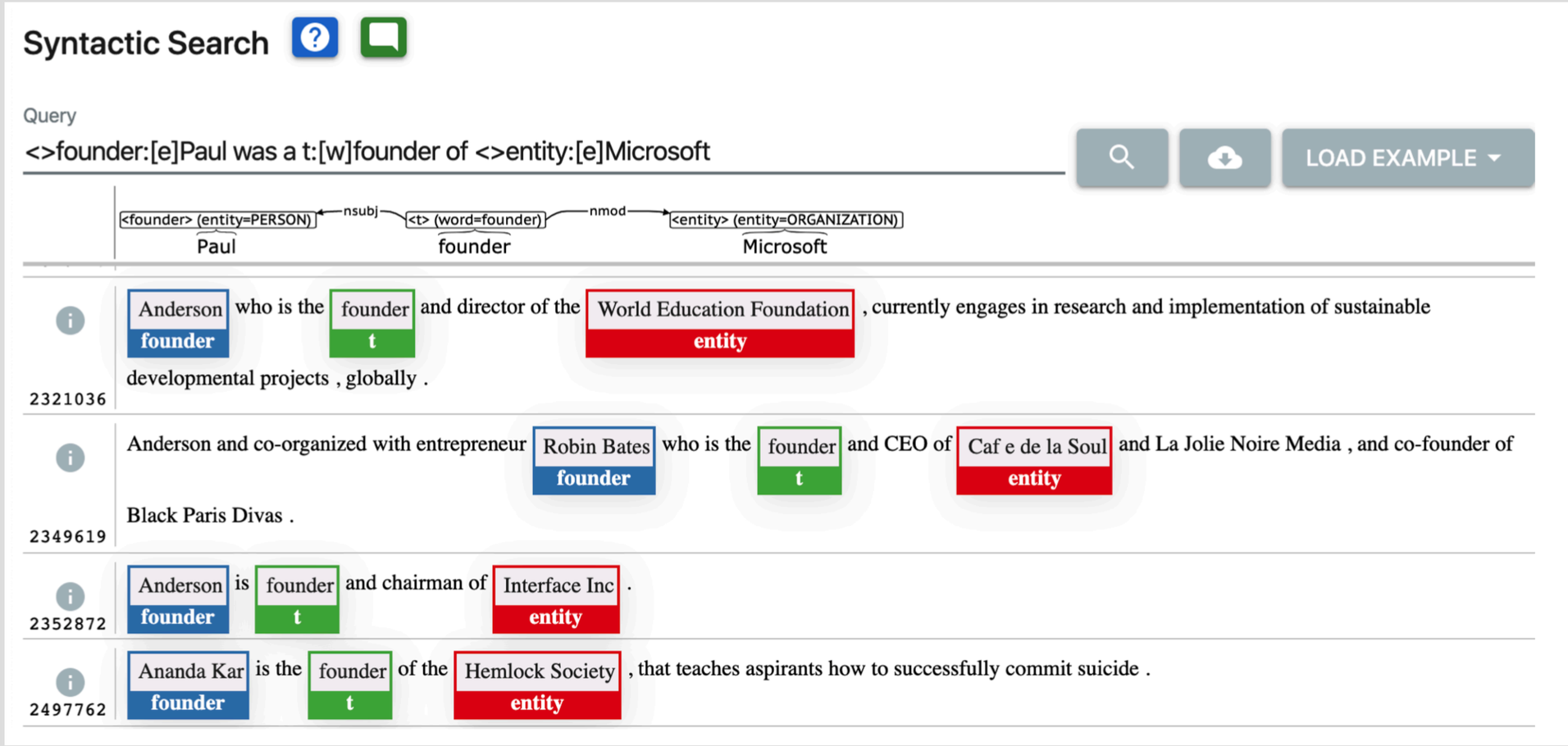




Strubell, Emma, et al. "Linguistically-informed self-attention for semantic role labeling." *EMNLP 2018*.

Example 2: Searching over Parsed Corpora

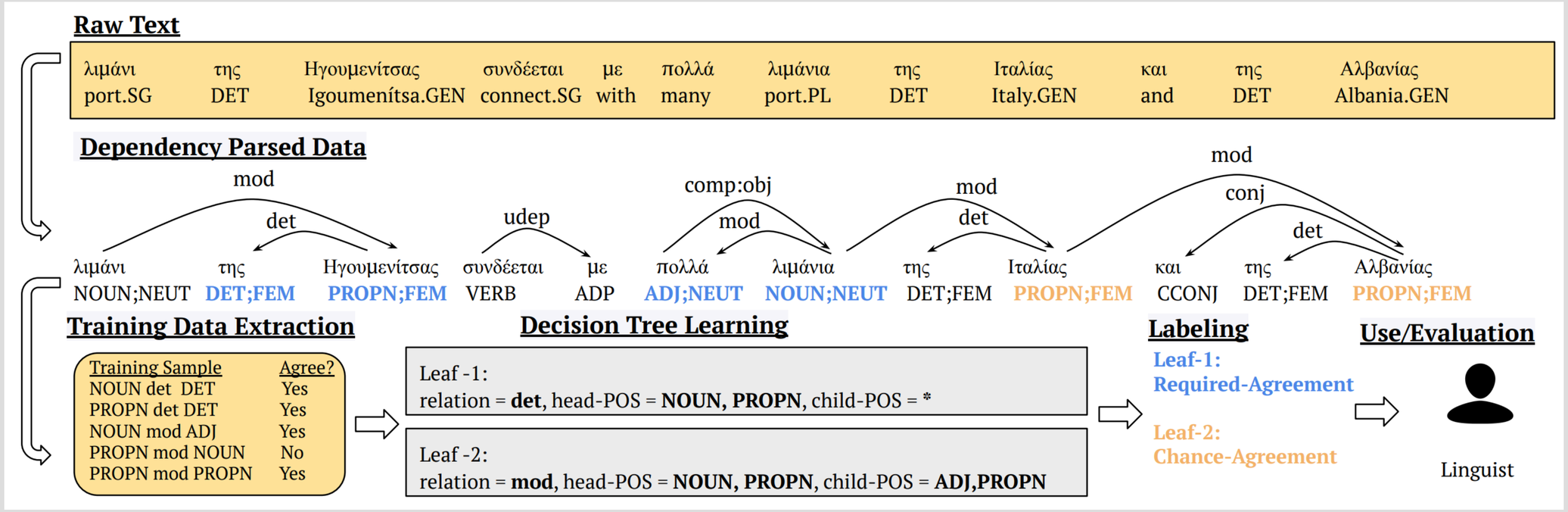
- Search using "syntactic regex"



Shlain, Micah, et al. "Syntactic Search by Example." *arXiv preprint arXiv:2006.03010* (2020).

Example 3: Understanding Language Structure

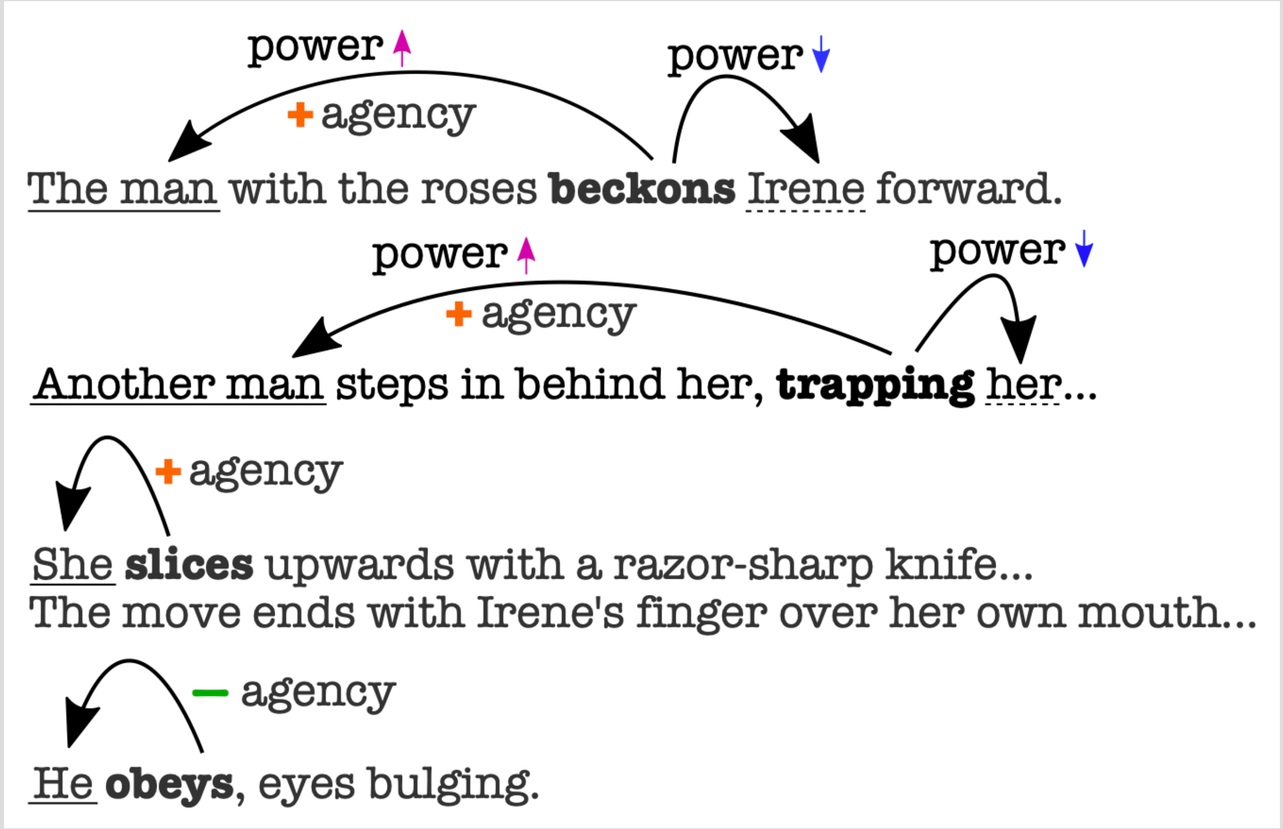
- Example of extracting morphological agreement rules using dependency relations



Chaudhary, Aditi, et al. "Automatic Extraction of Rules Governing Morphological Agreement." *EMNLP 2020*.

Example 4: Analysis of Other Linguistic Phenomena

- Examining power and agency in film scripts



Sap, Maarten, et al. "Connotation frames of power and agency in modern films." *EMNLP 2017*.

Parsing

- Predicting linguistic structure from input sentence
- **Transition-based models**
  - step through actions one-by-one until we have output
  - modeling “what transition action to take in each step”
- **Graph-based models**
  - calculate probability of each edge/constituent, and perform some sort of dynamic programming
  - modeling “what edge to build for connecting the constituents”

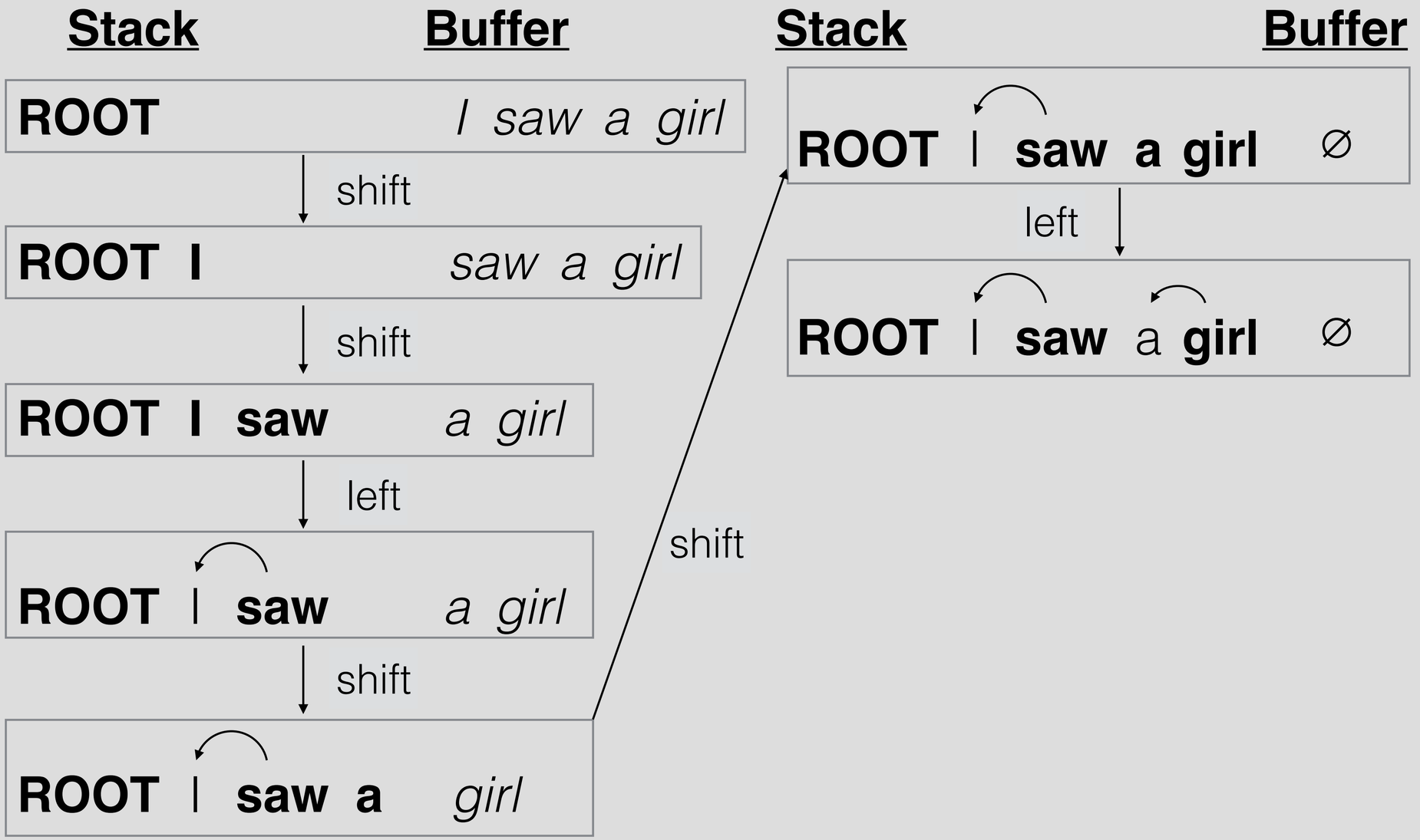
Shift-reduce Parsing

Arc Standard Shift-Reduce Parsing

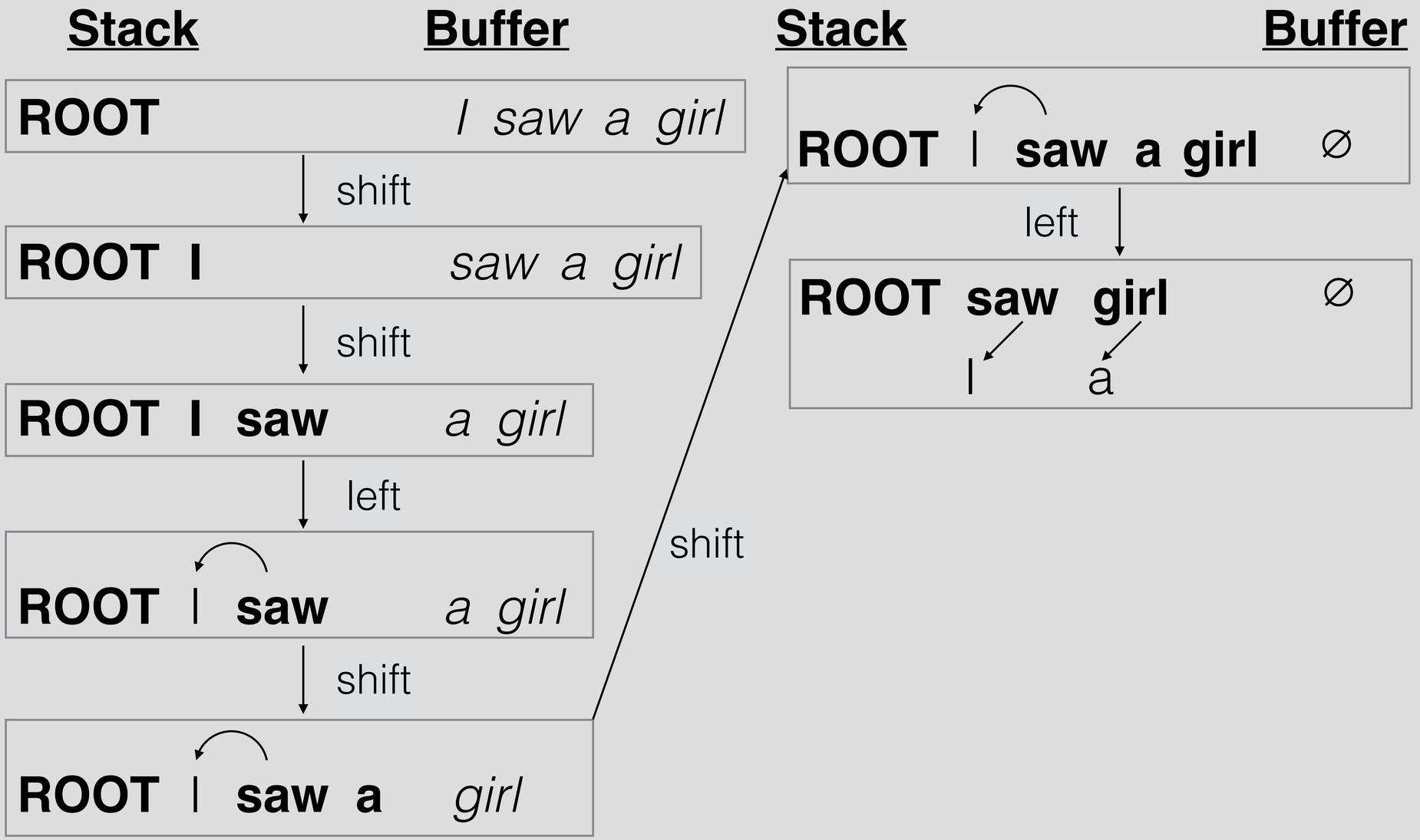
(Yamada & Matsumoto 2003, Nivre 2003)

- Process words one-by-one left-to-right
- Two data structures
  - **Queue/buffer:** of unprocessed words
  - **Stack:** of partially processed words
- At each point choose
  - **shift:** move one word from queue to stack
  - **reduce left:** top word on stack is head of second word
  - **reduce right:** second word on stack is head of top word
- Learn how to choose each action with a classifier

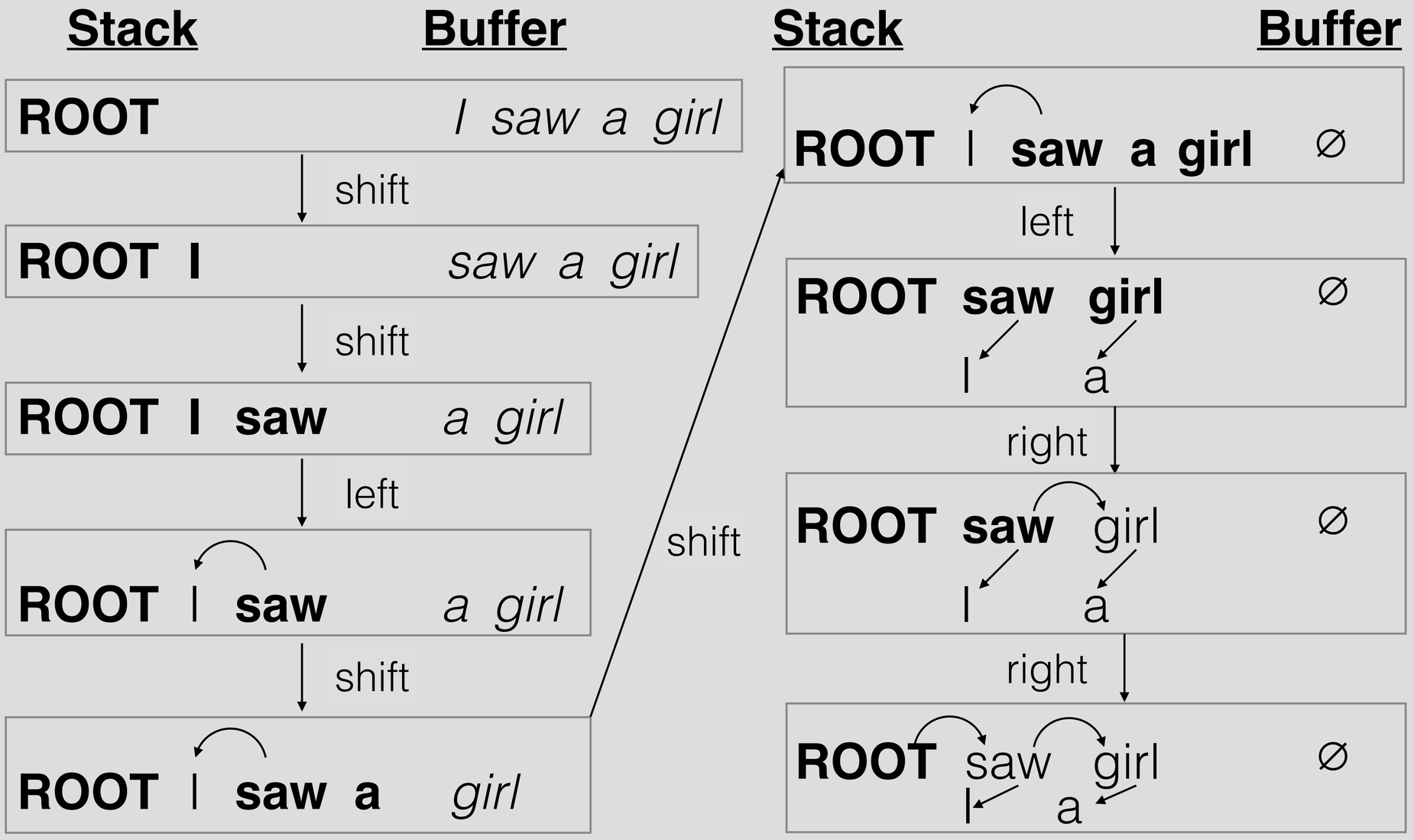
Shift Reduce Example



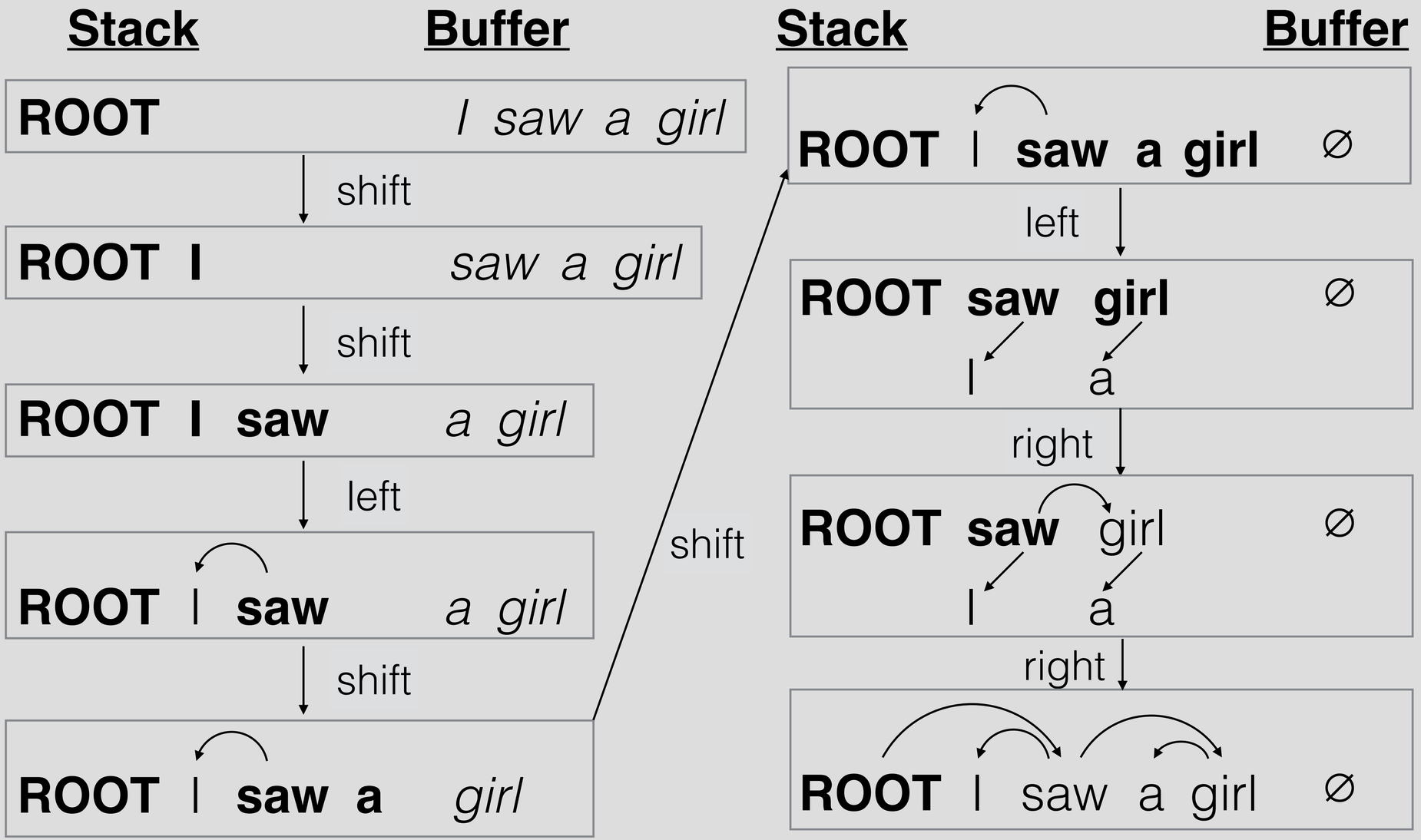
Shift Reduce Example



Shift Reduce Example



Shift Reduce Example



Classification for Shift-reduce

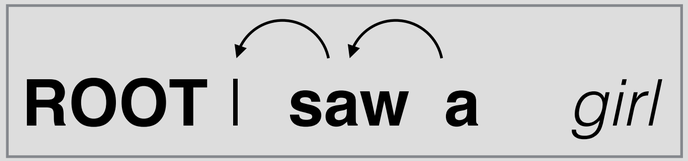
- Given a **configuration**

- Which **action** do we choose?

shift

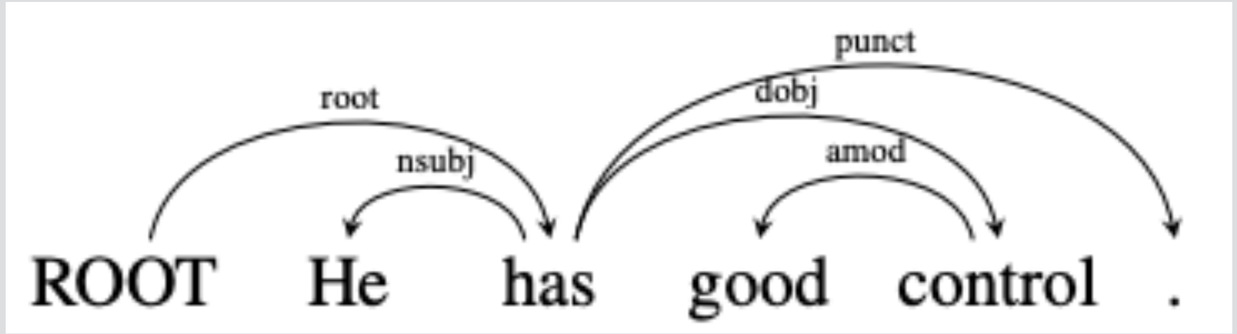
right

left



What Actions to Choose?

- Building a multi-class classifier
  - num_classes = 2 x num_arc_types (left/right reduce with a certain arc type) + 1 (shift)



A Fast and Accurate Dependency Parser using Neural Networks, Chen and Manning 2014

What Actions to Choose?

- Building a multi-class classifier
  - num_classes = 2 x num_arc_types (left/right reduce with a certain arc type) + 1 (shift)
- What features?

s_i: the i-th top element on the stack;
w: word token;
t: POS tag;
b_i: the i-th element on the buffer.

Single-word features (9)
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$;
$s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$
Word-pair features (8)
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$;
$s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$
$s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$
Three-word feaures (8)
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$;
$s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$;
$s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$;
$s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$

A Fast and Accurate Dependency Parser using Neural Networks, Chen and Manning 2014

What Actions to Choose?

- Building a multi-class classifier
  - num_classes = 2 x num_arc_types (left/right reduce with a certain arc type) + 1 (shift)
- What features?
  - Issues with conventional feature-engineering approaches:

    Sparsity, incompleteness (in terms of feature space), expensive feature computation (as the parsing runs dynamically and you have to look up the feature in a huge feature table)

A Fast and Accurate Dependency Parser using Neural Networks, Chen and Manning 2014

What Actions to Choose?

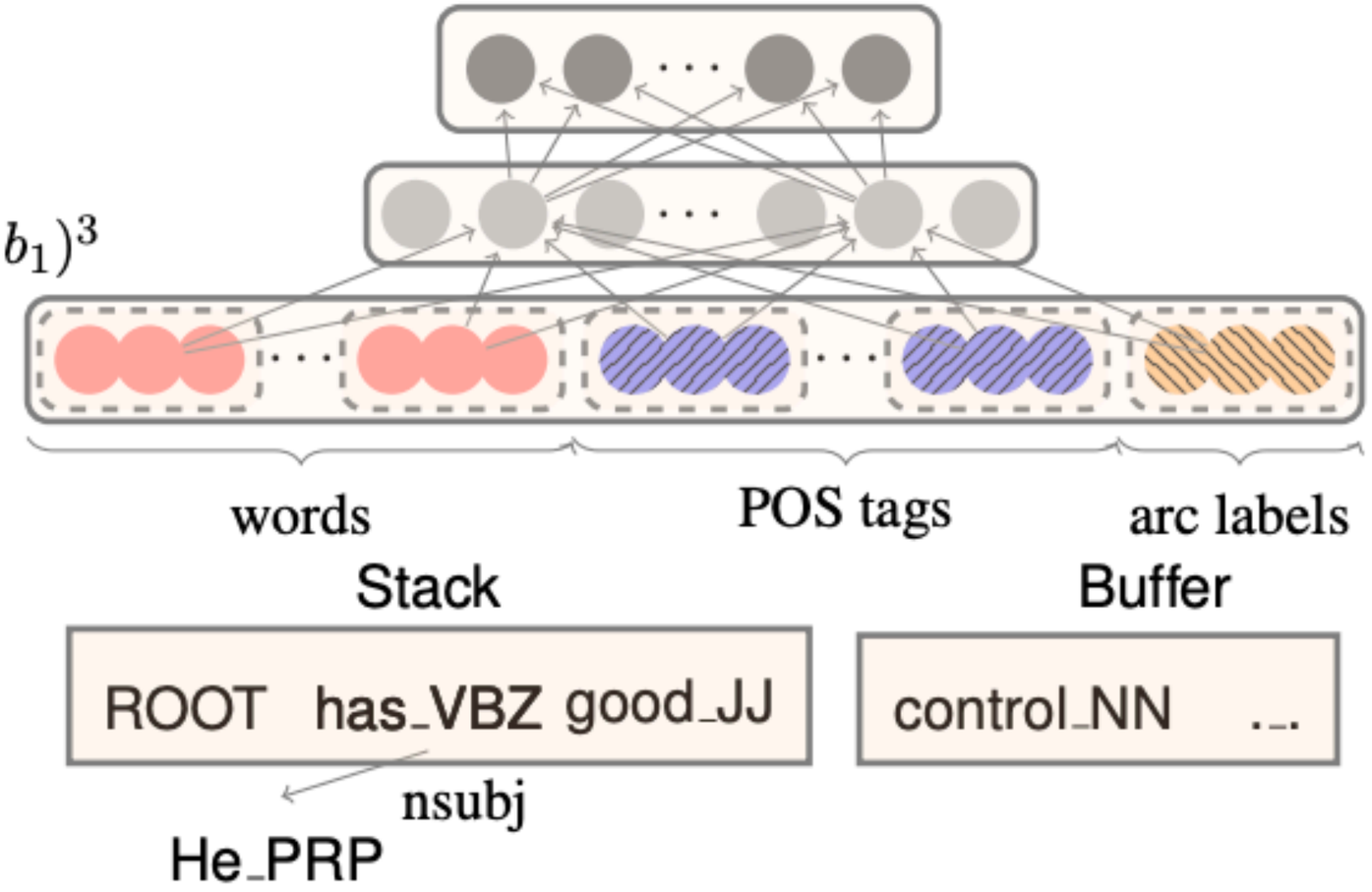
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

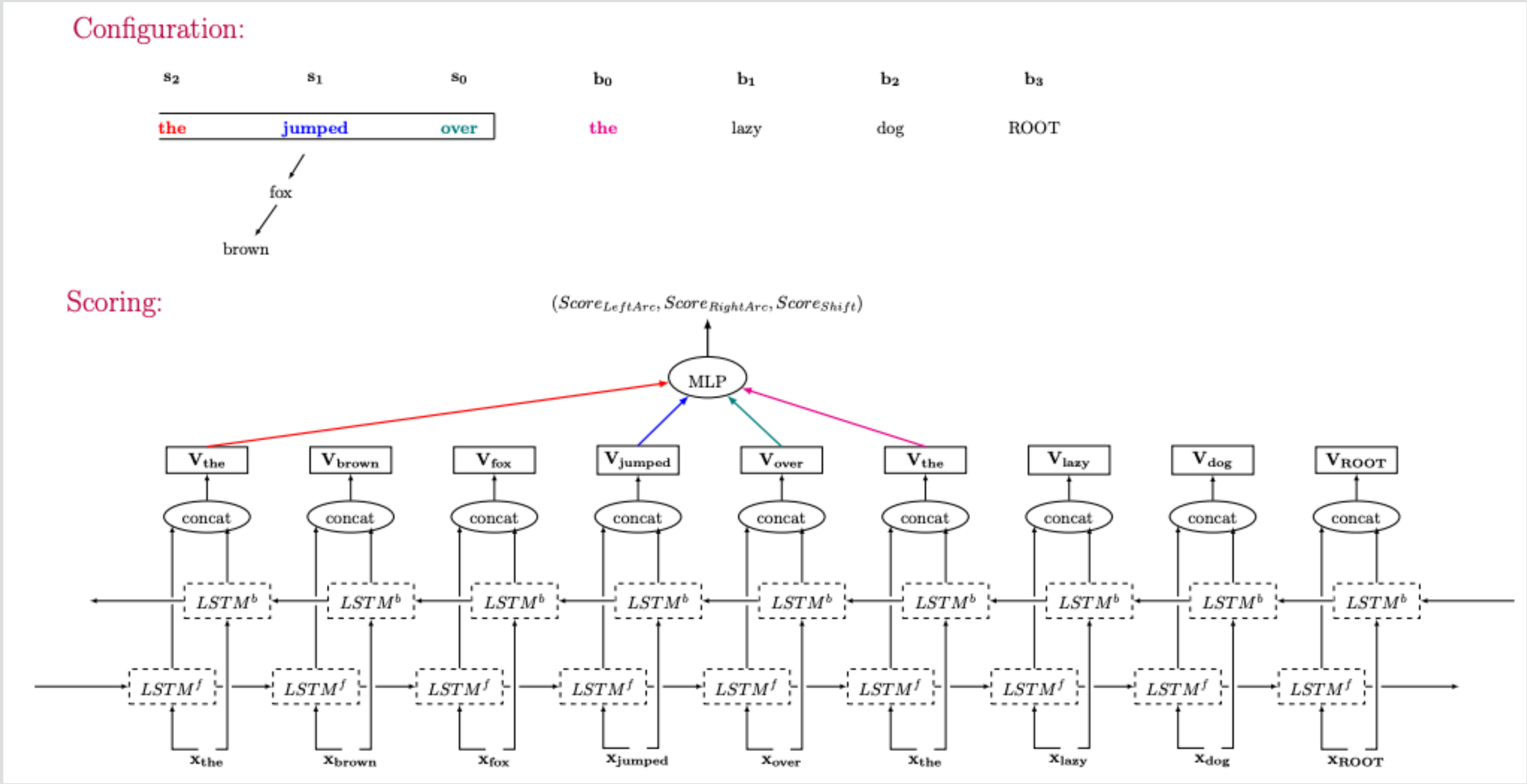
Input layer: $[x^w, x^t, x^l]$



Configuration

A Fast and Accurate Dependency Parser using Neural Networks, Chen and Manning 2014

What Actions to Choose?

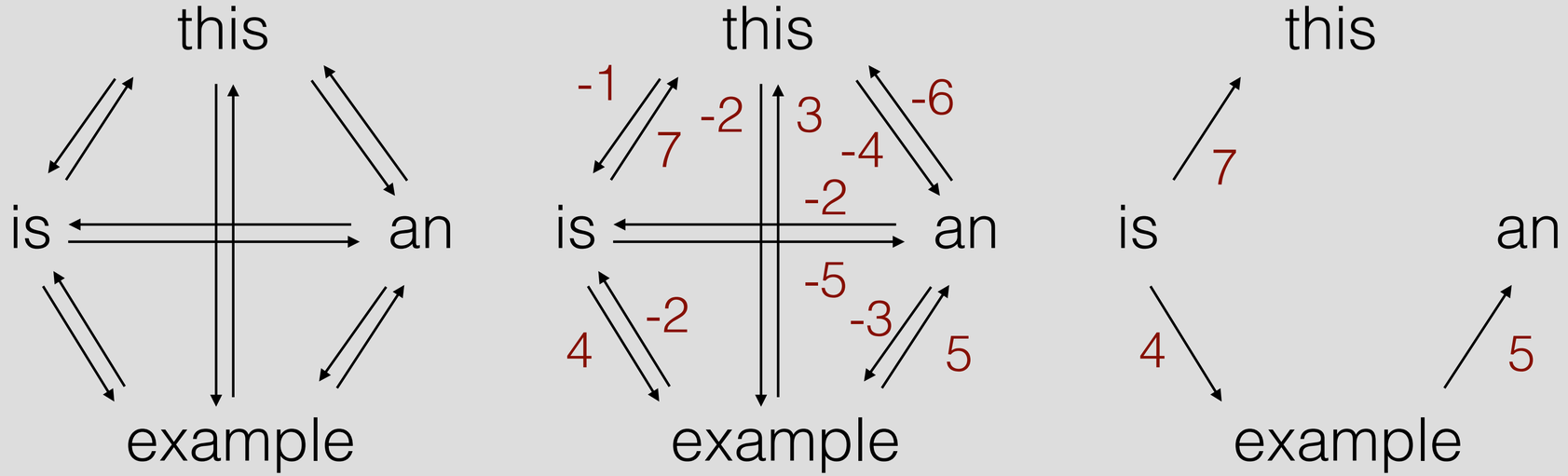


Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations, Kiperwasser and Goldberg, 2016

Graph-based Parsing

Graph-based Dependency Parsing

- Express sentence as fully connected directed graph
- Score each *edge* independently
- Find *maximal spanning tree*

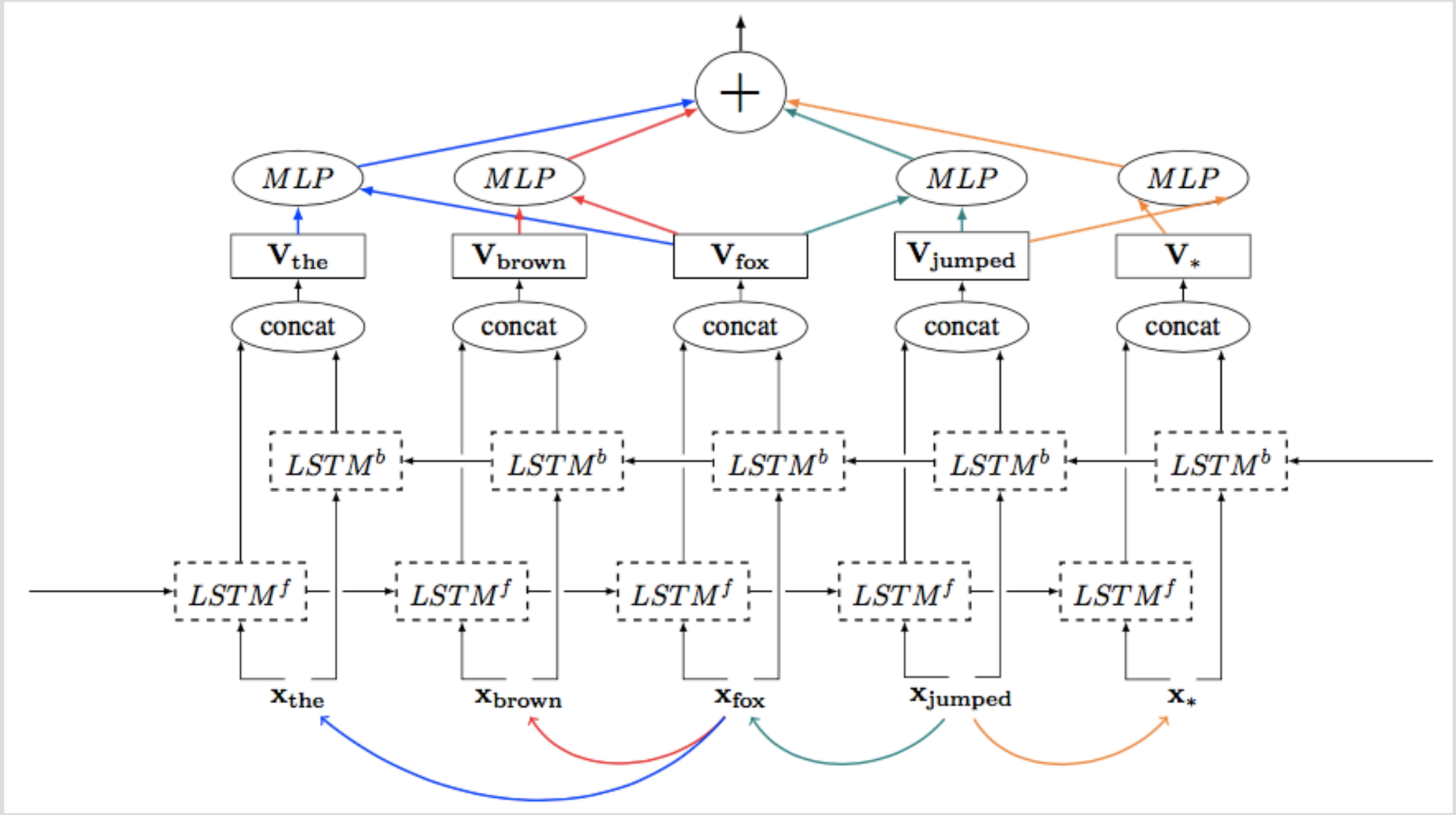


How to Score the Edge?

- Words in sentence x, tree T is a collection of directed edges (parent(i), i) for each word i
  - Parsing = identify parent(i) for each word
  - Each word has exactly one parent
- Log-linear CRF (discriminative): $P(T|\mathbf{x}) = \exp\left(\sum_i w^\top f(i, \text{parent}(i), \mathbf{x})\right)$
- Example of a feature = I[head=to & modifier=house]

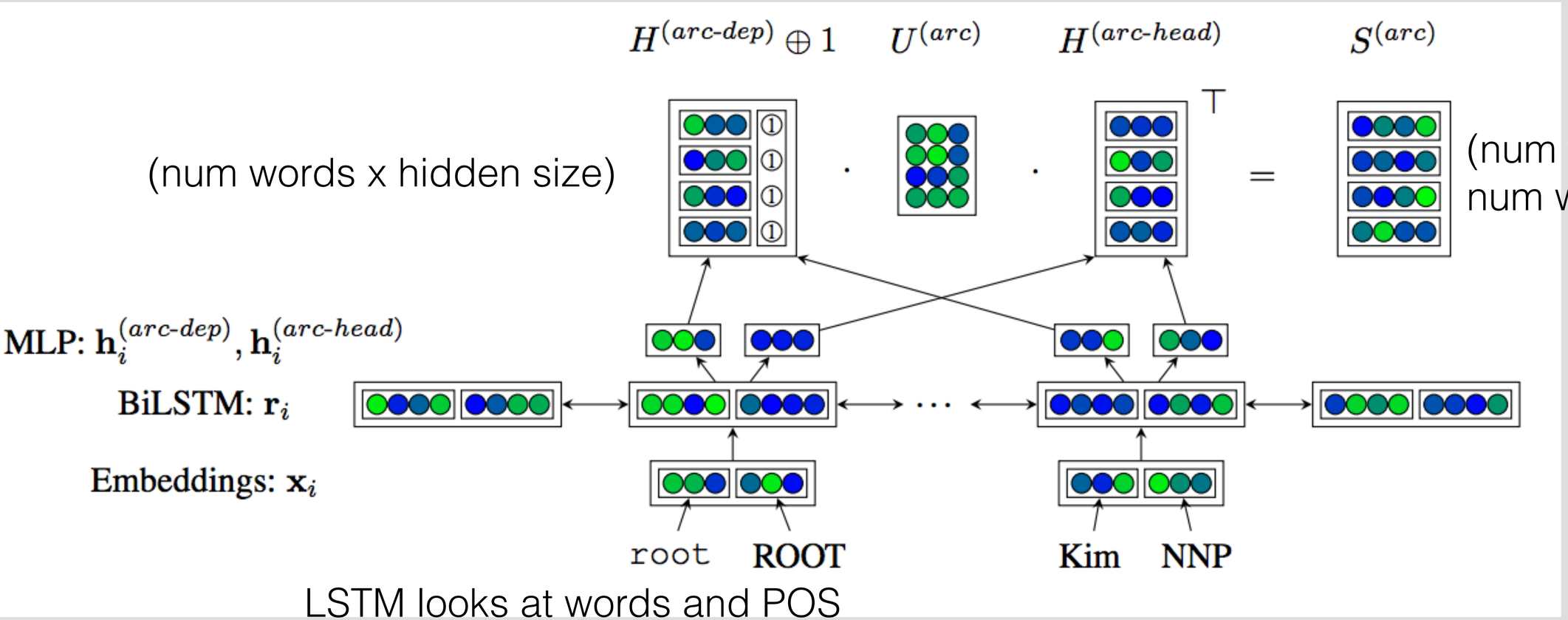
Sequence Model Feature Extractors

(Kipperwasser and Goldberg 2016)



Biaffine Neural Parsing

- Neural CRFs for dependency parsing: let *c* = LSTM embedding of *i*, *p* = LSTM embedding of *parent(i)*. *score*(*i*, *parent(i)*, *x*) = $p^T U c$



LSTM looks at words and POS

Deep Biaffine Attention For Neural Dependency Parsing, Dozat and Manning, 2017

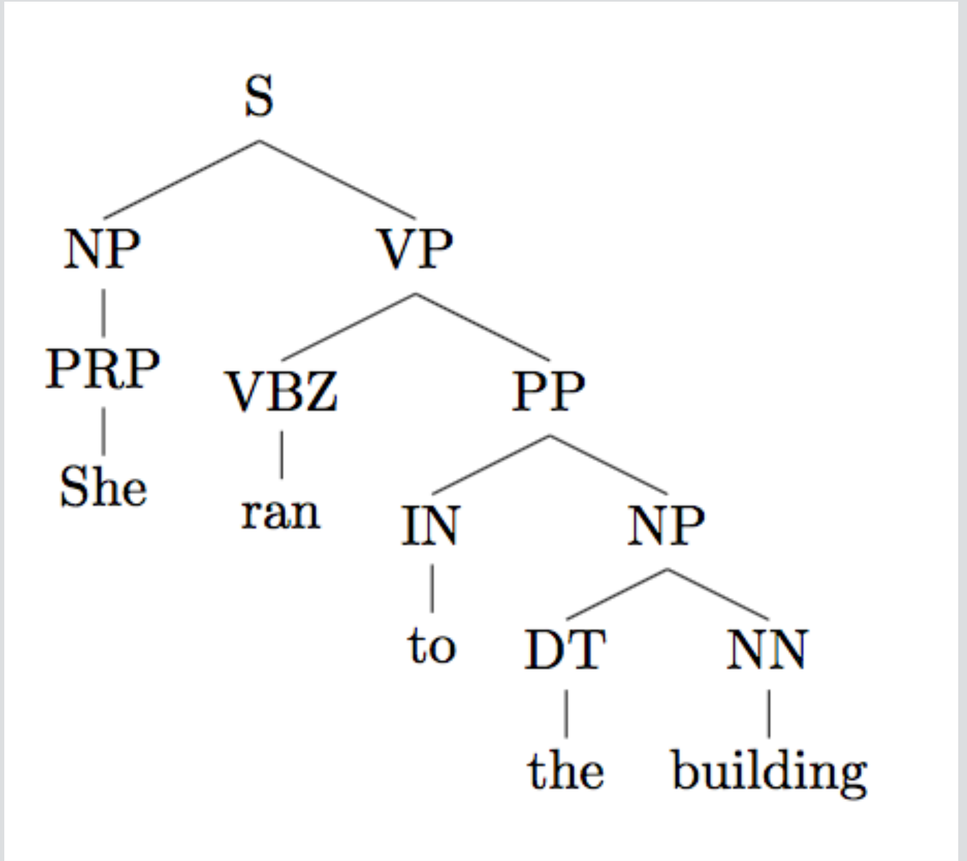
Graph-based vs. Transition Based

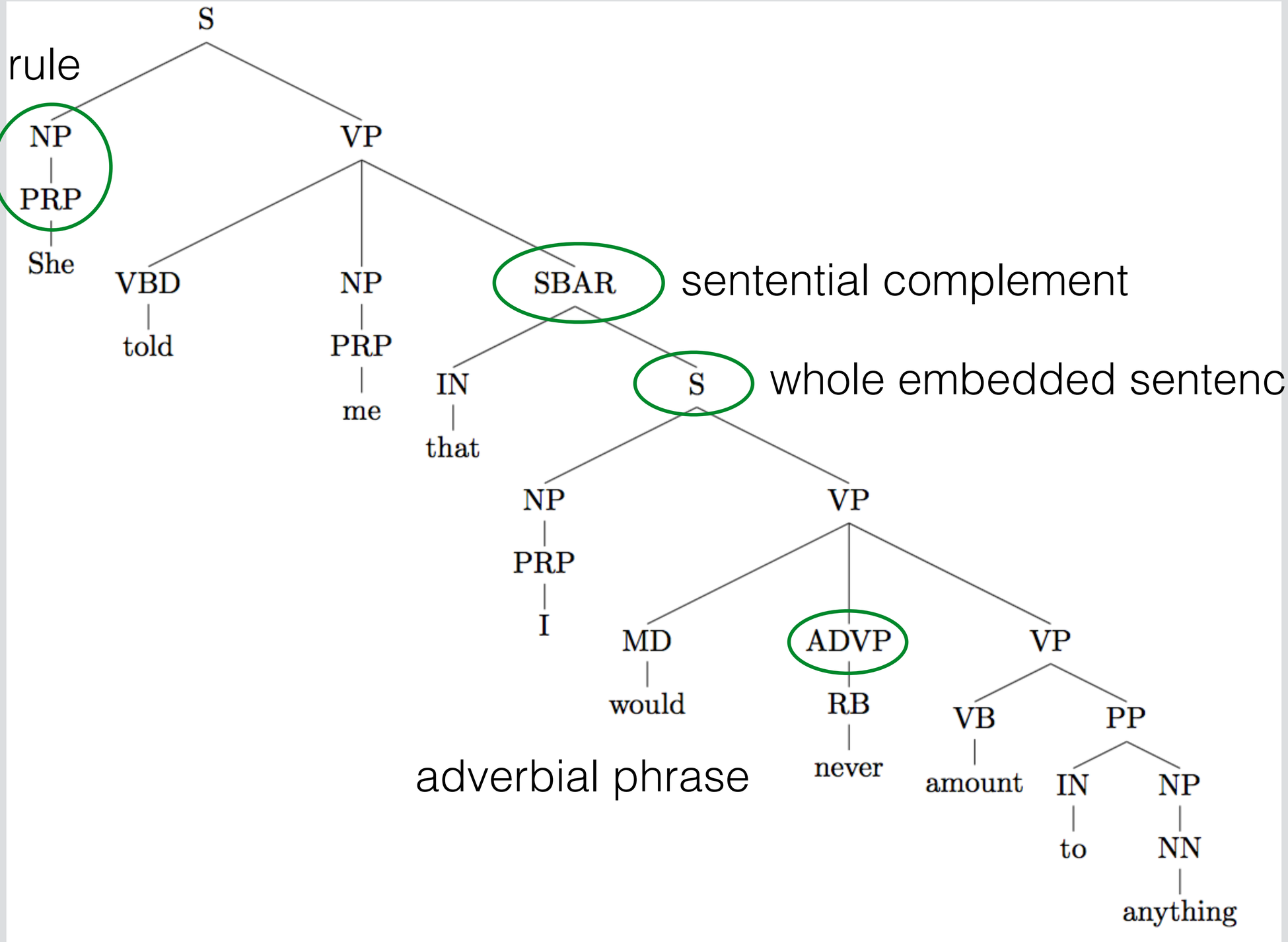
- **Transition-based**
  - + Easily condition on infinite tree context (structured prediction)
  - - Greedy search algorithm causes short-term mistakes
- **Graph-based**
  - + Can find exact best global solution via DP algorithm
  - - Have to make local independence assumptions

Syntactic Parsing 2: Constituency Parsing

Constituency Parsing

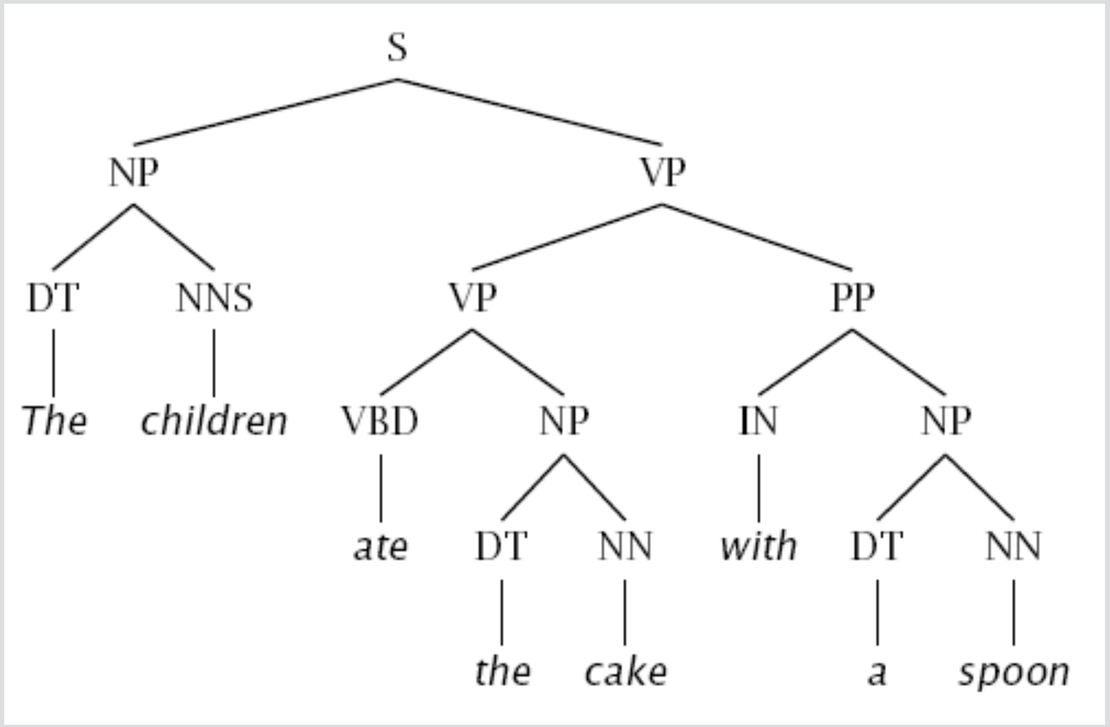
- Tree-structured syntactic analyses of sentences
- Common things: noun phrases, verb phrases, prepositional phrases
- Bottom layer is POS tags
- Examples will be in English. Constituency makes sense for a lot of languages but not all

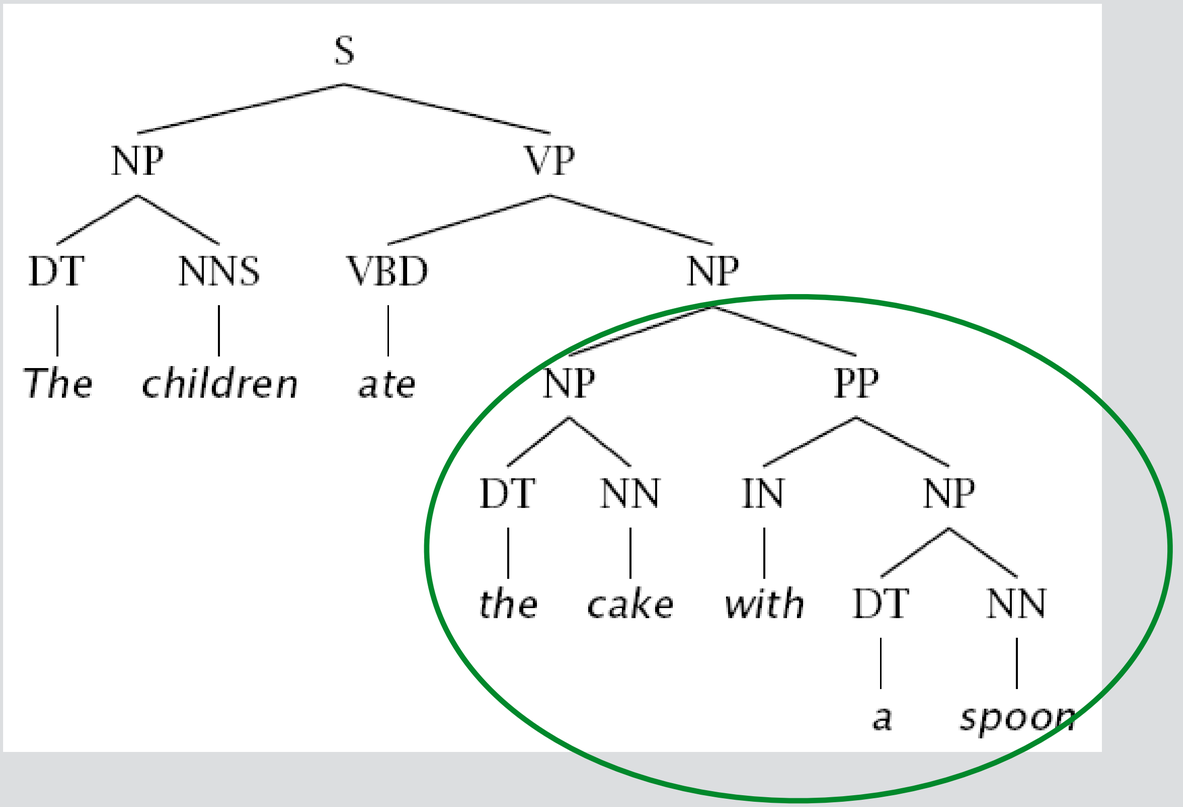



unary rule
S
NP
VP
PRP
She
VBD
NP
SBAR
sentential complement
told
PRP
IN
S
whole embedded sentence
me
that
NP
VP
PRP
I
MD
ADVP
VP
would
RB
VB
PP
adverbial phrase
never
amount
IN
NP
to
NN
anything

Challenges

- PP (Prepositional Phrase) attachment





same parse as “the cake with some icing”

Challenges

- Modifier scope:
  - e.g., plastic cup holder
- Complement structure:
  - e.g., The students complained to the professor that they didn’t understand
- Coordination scope:
  - e.g., The man picked up his hammer and *saw (verb or noun?)*

Context-Free Grammars, CKY

CFGs and PCFGs

Grammar (CFG)

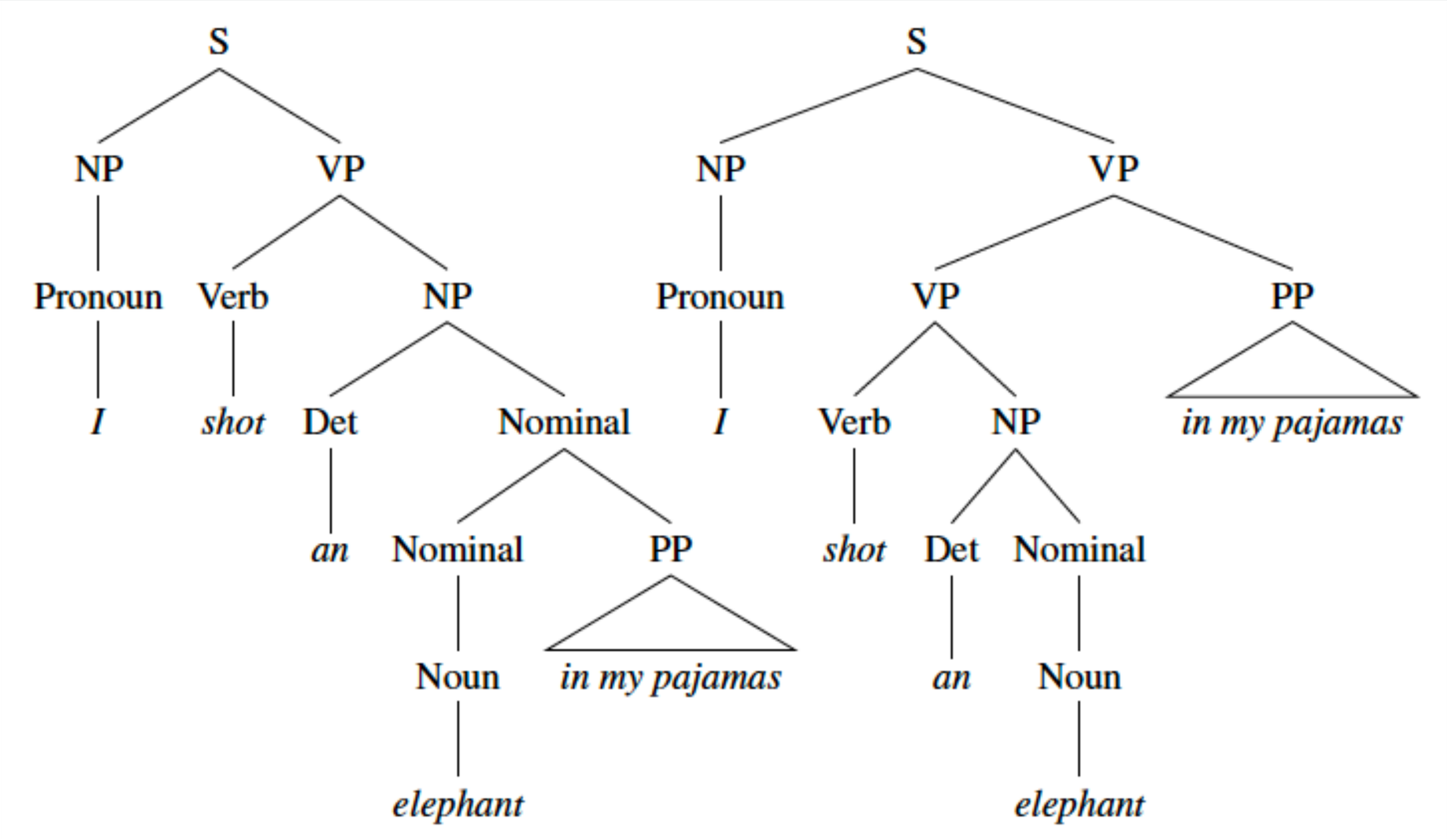
ROOT → S	NP → NP PP
S → NP VP	VP → VBP NP
NP → DT NN	VP → VBP NP PP
NP → NN NNS	PP → IN NP

Lexicon

NN → interest
NNS → raises
VBP → interest
VBZ → raises

- **Context-free grammar (CFG):** symbols which rewrite as one or more symbols
- **Lexicon** consists of “preterminals” (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules

Ambiguity under the same set of grammar and lexicon, e.g.,



Which one is more likely? Modeling the “probability”!

CFGs and PCFGs

Grammar (CFG)

ROOT → S	1.0	NP → NP PP	0.3		
S → NP VP	1.0	VP → VBP NP	0.7		
NP → DT NN	0.2	VP → VBP NP PP	0.3		
NP → NN NNS	0.5	PP → IN NP	1.0		

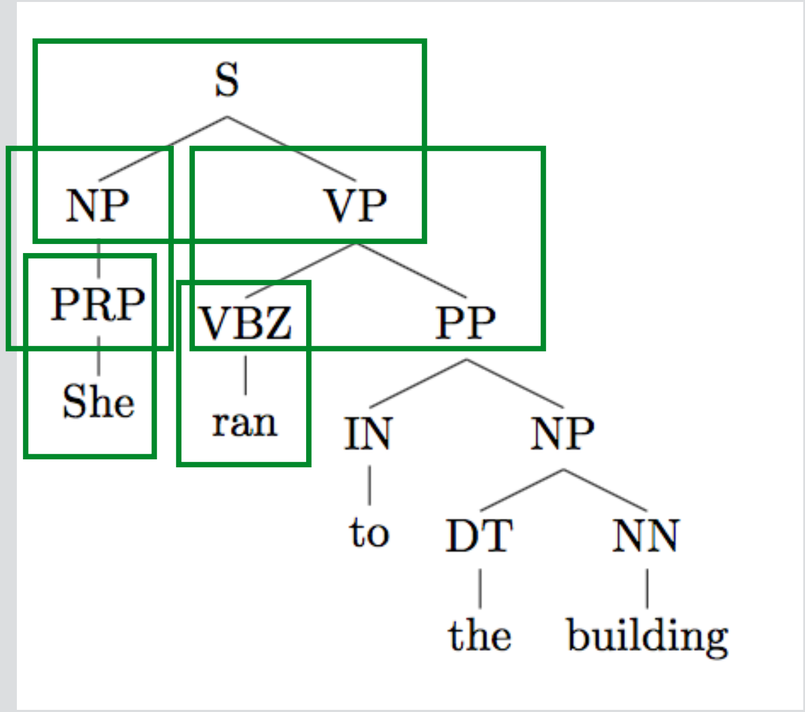
Lexicon

NN → interest	1.0		
NNS → raises	1.0		
VBP → interest	1.0		
VBZ → raises	1.0		

- Context-free grammar: symbols which rewrite as one or more symbols
- Lexicon consists of “preterminals” (POS tags) rewriting as terminals (words)
- CFG is a tuple (N, T, S, R): N = nonterminals, T = terminals, S = start symbol (generally a special ROOT symbol), R = rules
- **PCFG**: probabilities associated with rewrites, normalize by source symbol

PCFGs

- Tree *T* is a series of rule applications *r*. $P(T) = \prod_{r \in T} P(r|\text{parent}(r))$



$$P(T) = P(S \rightarrow NP\ VP) \times P(NP \rightarrow PRP) \times P(PRP \rightarrow She) \times P(VP \rightarrow VBZ\ PP) \times P(VBZ \rightarrow ran) \times P(PP \rightarrow IN\ NP) \times P(IN \rightarrow to) \times P(NP \rightarrow DT\ NN) \times P(DT \rightarrow the) \times P(NN \rightarrow building)$$

Learning PCFG: how do I know the probability for each derivation?

Using PCFG: how do I apply PCFG to parse a new sentence into a constituency parse tree?

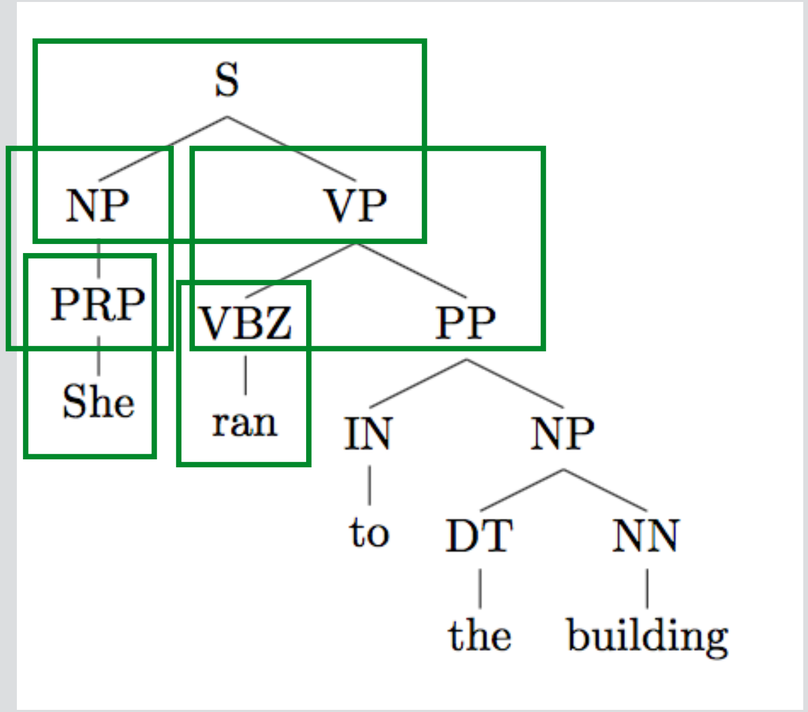
Learning a PCFG

- Given a set of example trees (from a treebank), the underlying CFG can simply be all production rules seen in the corpus
- Learning goal: estimate the probability of each production rule
  - e.g., *P(NP -> PRP)* or *P(PRP | NP)* describes the probability of applying the rule "NP -> PRP" when the current (parent node) is "NP"
- Maximum likelihood estimation (similar as HMMs')

$$P(\alpha -> \beta) = \frac{Count(\alpha -> \beta)}{Count(\alpha)}$$

PCFGs

- Tree *T* is a series of rule applications *r*. $P(T) = \prod_{r \in T} P(r|\text{parent}(r))$



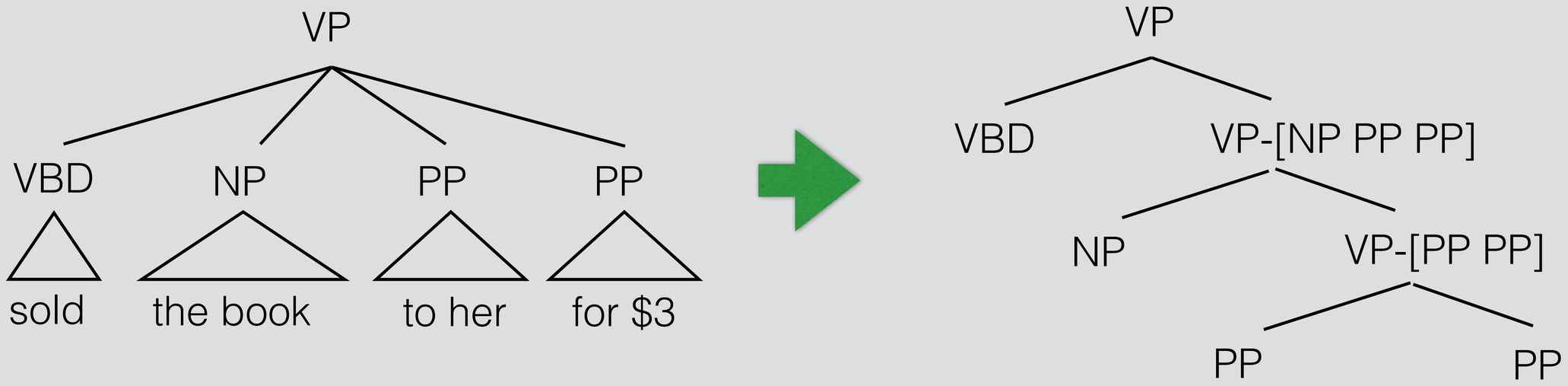
$$P(T) = P(S \rightarrow NPVP) \times P(NP \rightarrow PRP) \times P(PRP \rightarrow She) \times P(VP \rightarrow VBZ\ PP) \times P(VBZ \rightarrow ran) \times P(PP \rightarrow INNP) \times P(IN \rightarrow to) \times P(NP \rightarrow DT\ NN) \times P(DT \rightarrow the) \times P(NN \rightarrow building)$$

Learning PCFG: MLE by counting + normalization, similar to learning HMMs

Using PCFG: $\text{argmax}_T P(T)$

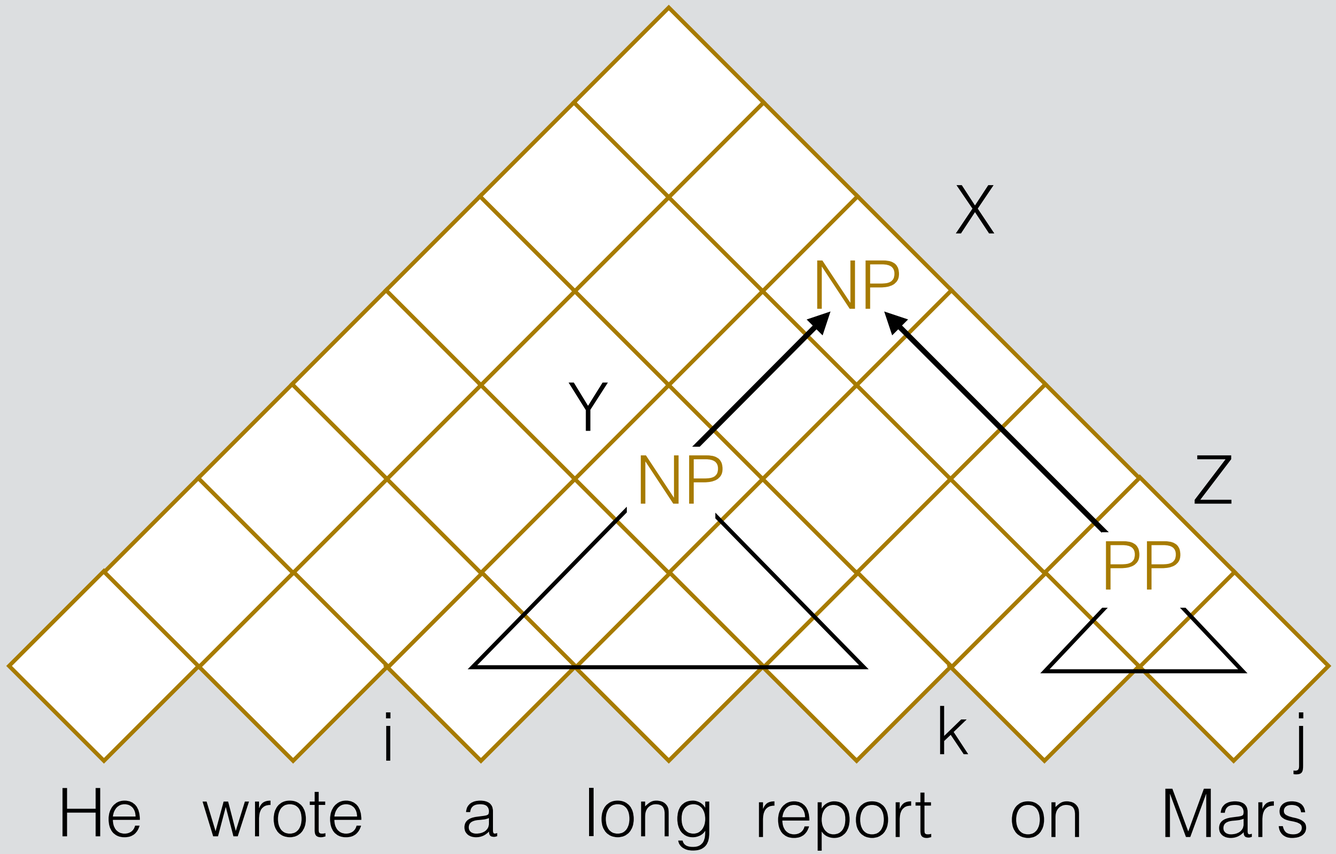
Binarization

- To parse efficiently, we assume PCFGs in **Chomsky Normal Form (CNF)**, i.e., having either two (nonterminal) subtree children or one (terminal) leaf value.



Inference: CKY

- Find $\text{argmax}_T P(T)$
- Dynamic programming: chart maintains the *best* way of building **symbol X** over **span (i, j)**
- CKY is like Viterbi



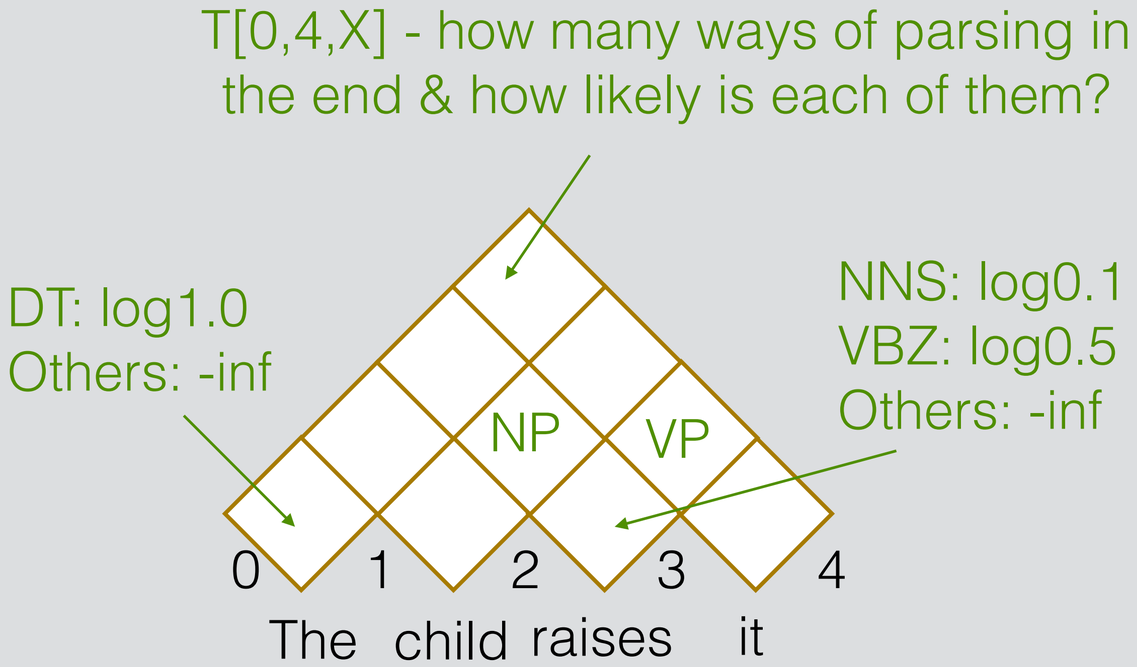
Cocke-Kasami-Younger

CKY

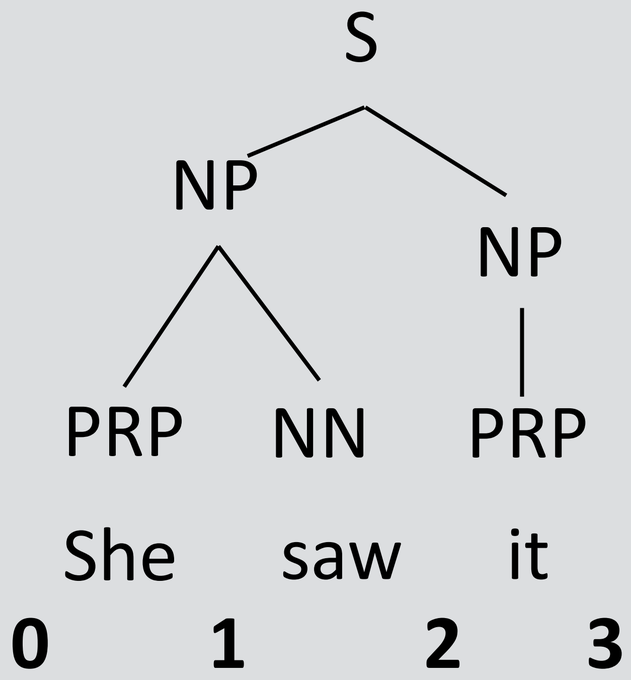
- Chart: T[i,j,X] = *best* score for X (a POS tag) over (i, j)
- Base: T[i,i+1,X] = log P(X → w_i)
- Loop over all split points k, apply rules X -> Y Z to build X in every possible way
- Recurrence:

$$T[i,j,X] = \max_{r: X \rightarrow X1\ X2} \max_{k} T[i,k,X1] + T[k,j,X2] + \log P(X \rightarrow X1\ X2)$$

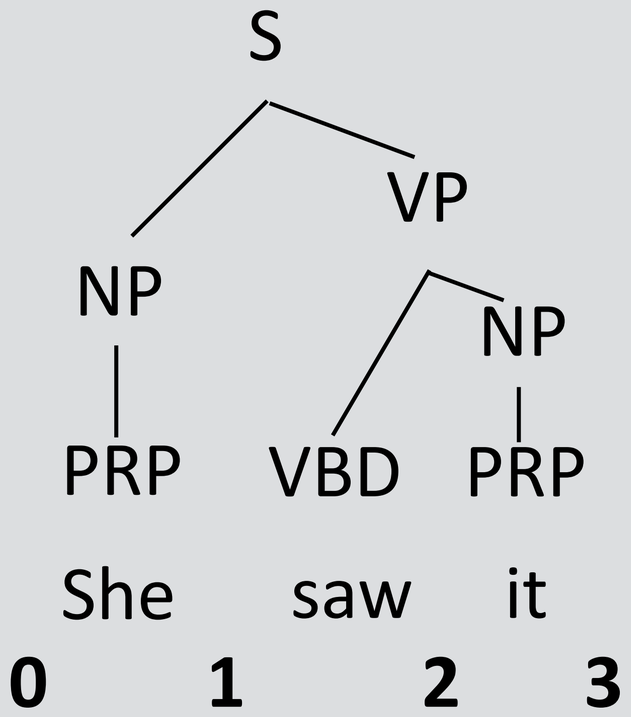
- Runtime: $O(n^3G)$ G = grammar constant



Parser Evaluation



S(0,3),
NP(0,2),
NP(2,3)



S(0,3),
NP(0,1),
VP(1,3),
NP(2,3)

- Precision: number of correct brackets / num pred brackets = 2/3
- Recall: number of correct brackets / num of gold brackets = 2/4
- F1: harmonic mean of precision and recall = $(1/2 * ((2/4)^{-1} + (2/3)^{-1}))^{-1}$
  $= 0.57$

Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
  - Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best performance: ~96 F1

https://github.com/sebastianruder/NLP-progress

- Other languages: results vary widely depending on annotation + complexity of the grammar

Dependency vs. Constituency

- Dependency is often more useful in practice (models predicate argument structure)
- Slightly different representational choices:
  - PP attachment is better modeled under dependency
  - Coordination is better modeled under constituency
- Dependency parsers are easier to build: no “grammar engineering”, no unaries, easier to get structured discriminative models working well
- Dependency parsers are usually faster
- Dependencies are more universal cross-lingually