Sequence labeling

CMSC 723 / LING 723 / INST 725

Hal Daumé III [he/him]

15 Oct 2019

(many slides c/o Marine Carpuat or Graham Neubig)

Announcements, logistics

- Projects:
  - Google cloud $$$ available if you need it ($50/student)
  - Project pitches *one week* from today!
    - One slide (pdf, no animations), two minutes!
    - One page write-up
    - Due by NOON (so I have time to assemble them)
    - You will present in a (pseudo-)random order
- HW3 due Thursday (before class)

Last time

- Language modeling as a prediction problem
- Feed-forward neural language models (& embeddings)
- Recurrent neural language models

Today

- Sequence labeling as independent predictions
- Structured perceptron for sequence labeling
- Do we really need structured features?
- Recurrent neural network taggers

Sequence labeling is a workhorse of NLP

- NLP for NLP's sake:
  - Part of speech tagging
  - Word segmentation
  - Morphological segmentation
  - All-word word sense disambiguation
  - …even parsing!
- NLP for other's sake:
  - Named entity recognition
  - Grammatical error detection
  - Spell correction
  - …
- General constraint: 1-1 mapping between input to
  - Plus "BIO" trick for encoding subsequence problems

x

y

ກຸ່ມຊາດພັນໄທ-ກ
ບໍລິສຸດແລະເປັນຕົ
ສະດີດັ່ງກ່າວຖືກຂື

uyuyorum	I am sleeping
uyuyorsun	you are sleeping
uyuyor	he/she/it is sleeping
uyuyoruz	we are sleeping
uyuyorsunuz	you are sleeping
uyuyorlar	they are sleeping
uyuduk	we slept
uyudukça	as long as (somebody) sleeps
	ve must sleep
	vithout sleeping
	our sleeping
	vhile (somebody) is sleeping
	vhen (somebody) sleeps
	o cause somebody to sleep
	o cause (somebody) to cause (another) to sleep
uyutturtturmak	to cause (somebody) to cause (some other) to cause (yet another) to sleep

Allergiantikropparna känner igen det ämne man är allergisk mot, till
exempel pollen. När man andas in pollen sätts en allergisk reaktion igång
och olika ämnen, bland annat histamin, frigörs. När histamin och andra
ämnen frisätts vid den allergiska reaktionen startar en inflammation i
ögonen och näsans slemhinnor. Det går inte att stoppa kroppens
allergiska reaktioner helt och hållet, men mediciner kan dämpa besvären.
Genom att använda läkemedel ska man kunna leva som vanligt och vistas
utomhus trots att det finns pollen i luften. Man kan pröva nässprej,
ögondroppar eller tabletter mot allergi. Om man blir bättre av
medicinerna är det troligt att pollenallergi är orsaken. Besvären kan också
bero på en vanlig förkylning, och då hjälper inte medicinerna. Om man är
osäker kan det vara bra att fråga om råd på ett apotek eller besöka en
läkare. Ibland kan det hjälpa att bara skölja och rensa näsan från pollen

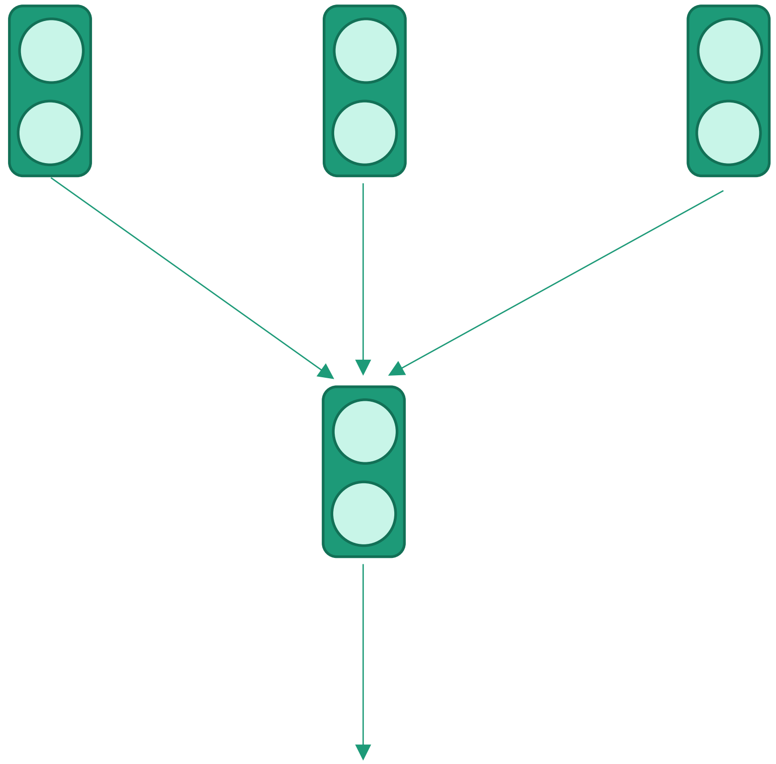
Sequence labeling as independent predictions

x = 日本語　が　あまり　話せません

y = noun　part　adv　verb

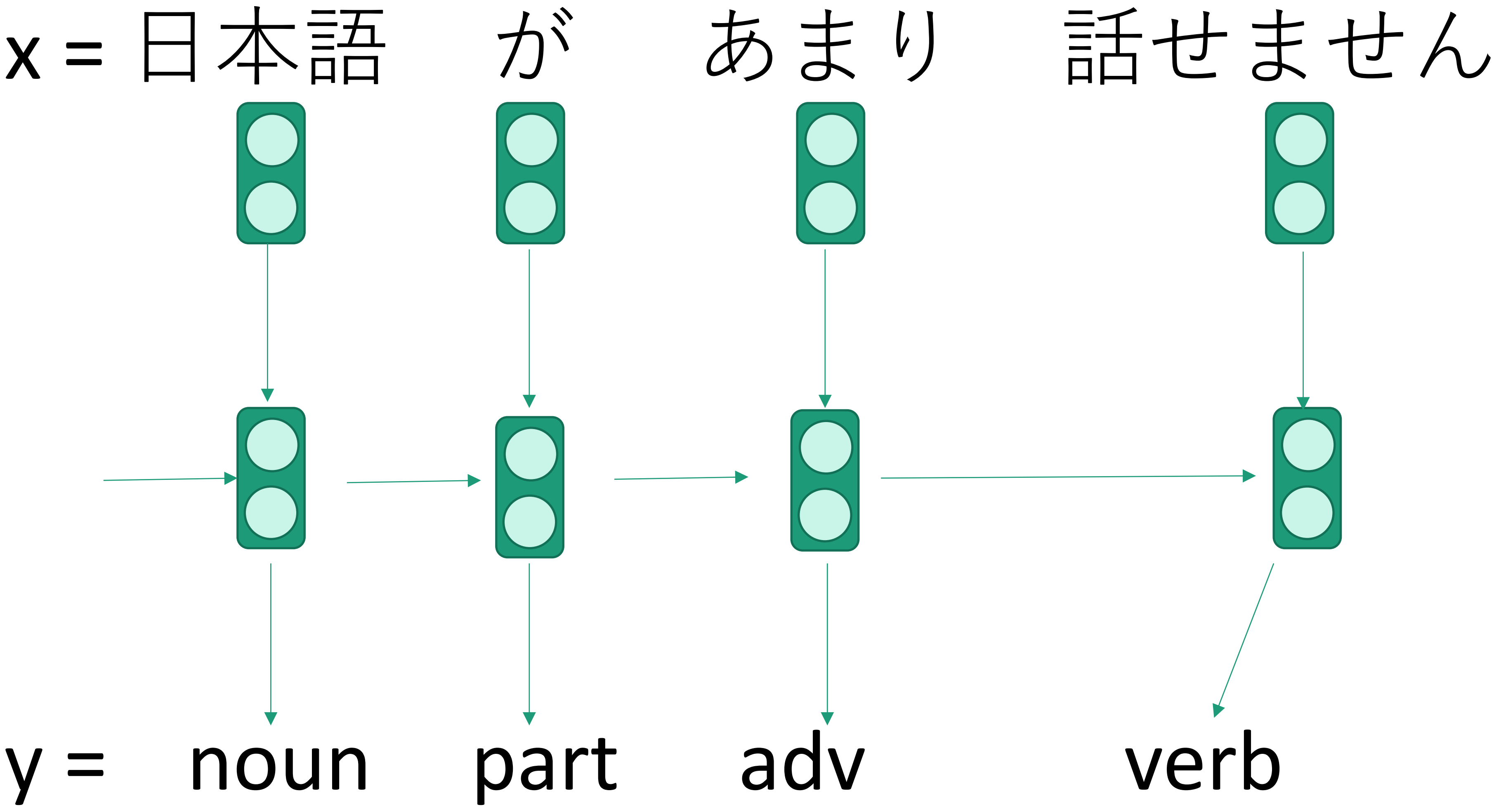
Sequence labeling as independent predictions

x = 日本語　が　あまり　話せません



y = noun　part　adv　verb

Sequence labeling as independent predictions



POS tagging
Sequence labeling with the perceptron

Sequence labeling problem

- Input:
  - sequence of tokens $x = [x_1 \ldots x_L]$
  - Variable length L
- Output (aka label):
  - sequence of tags $y = [y_1 \ldots y_L]$
  - # tags = K
  - Size of output space?

Structured Perceptron

- Perceptron algorithm can be used for sequence labeling
- But there are challenges
  - How to compute argmax efficiently?
  - What are appropriate features?
- Approach: leverage structure of output space

Reminder: multiclass perceptron

Algorithm 1 Perceptron learning algorithm

procedure PERCEPTRON($\boldsymbol{x}_{1:N}, y_{1:N}$)
    repeat
        Select an instance i
        $\hat{y} \leftarrow \arg\max_y \boldsymbol{\theta}_t^\top \boldsymbol{f}(\boldsymbol{x}_i, y)$
        if $\hat{y} \neq y_i$ **then**
            $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, \hat{y})$
        else
            do nothing
    until tired

$$\hat{y} = \arg\max_{y} \boldsymbol{\theta}^\top \mathbf{f}(\mathbf{x}, y)$$

Feature function representation

Weights

Solving the argmax problem for sequences with dynamic programming

$$\boldsymbol{x} = \text{“ monsters eat tasty bunnies “}$$
$$\boldsymbol{y} = \quad \text{noun verb adj noun}$$

- Efficient algorithms possible if **the feature function decomposes over the input**
- This holds for unary and markov features used for POS tagging

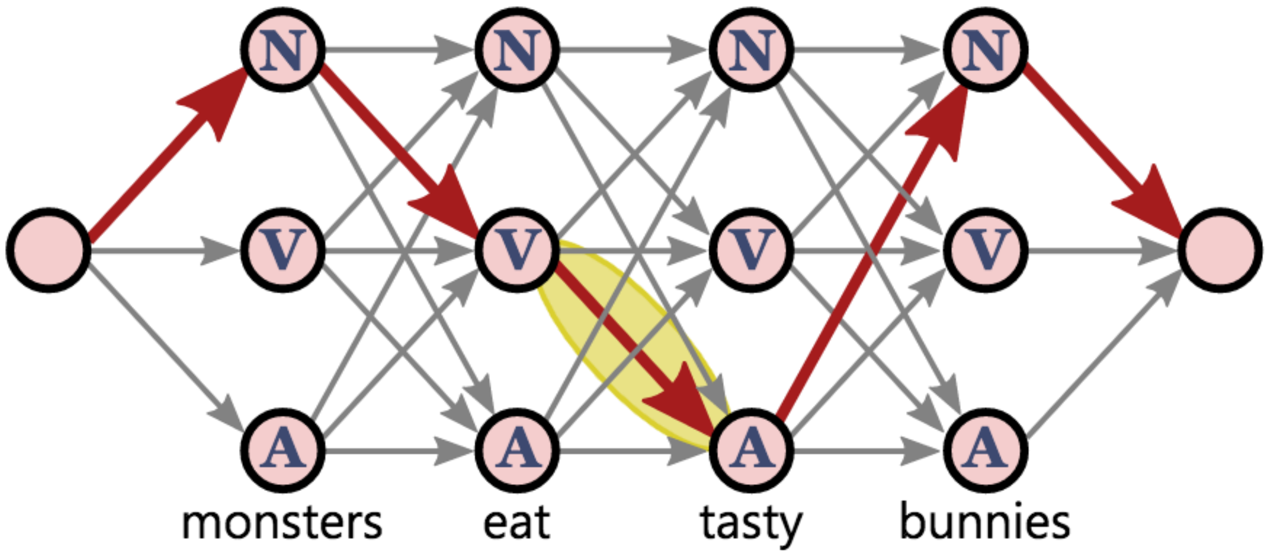
Feature functions for sequence labeling

$x =$ " monsters eat tasty bunnies "

$y =$ noun verb adj noun

- Standard features of POS tagging
  - **Unary features:** # times word w has been labeled with tag l for all words w and all tags l
  - **Markov features:** # times tag l is adjacent to tag l' in output for all tags l and l'
- Size of feature representation is constant wrt input length

Solving the argmax problem for sequences



- Trellis sequence labeling
  - Any path represents a labeling of input sentence
  - Gold standard path in red
  - Each edge receives a weight such that adding weights along the path corresponds to score for input/ouput configuration
- Any max-weight max-weight path algorithm can find the argmax
  - e.g. Viterbi algorithm $O(LK^2)$

Defining weights of edge in trellis

Unary features at position l together with Markov features that end at position l

$$w \cdot \phi(x, y) = w \cdot \sum_{l=1}^{L} \phi_l(x, y) \qquad \text{decomposition of structure} \qquad (17.35)$$

$$= \sum_{l=1}^{L} w \cdot \phi_l(x, y) \qquad \text{associative law} \qquad (17.36)$$

- Weight of edge that goes from time l-1 to time l, and transitions from y to y'

$$w \cdot \phi_l(x, \cdots \circ y \circ y')$$

Dynamic program

- Define: the score of best possible output prefix up to and including position l that labels the l-th word with label k

$$\alpha_{l,k} = \max_{\hat{y}_{1:l-1}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

- With decomposable features, alphas can be computed recursively

$$\alpha_{l+1,k} \quad = \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \right]$$

$$\alpha_{0,k} = 0 \quad \forall k \tag{17.41}$$

$$\zeta_{0,k} = \varnothing \quad \forall k \tag{17.42}$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \tag{17.43}$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \Big(\boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \Big) \tag{17.44}$$

distributive law over dot products

$$= \max_{\hat{y}_{1:l}} \Big[\boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \Big] \tag{17.45}$$

separate out final label from prefix, call it k’

$$= \max_{\hat{y}_{1:l-1}} \max_{k'} \Big[\boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k' \circ k) \Big] \tag{17.46}$$

swap order of maxes, and last term doesn’t depend on prefix

$$= \max_{k'} \Big[\Big[\max_{\hat{y}_{1:l-1}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') \Big] + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \Big] \tag{17.47}$$

apply recursive definition

$$= \max_{k'} \Big[\alpha_{l,k'} + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \Big] \tag{17.48}$$

Algorithm 42 $\textsc{ArgmaxForSequences}(\boldsymbol{x}, \boldsymbol{w})$

$L \leftarrow \textsc{len}(\boldsymbol{x})$

$\alpha_{l,k} \leftarrow 0, \quad \zeta_{k,l} \leftarrow 0, \quad \forall\, k = 1 \dots K, \quad \forall l = 0 \dots L$ // initialize variables

for $l = 0 \dots L\text{-}1$ **do**

  for $k = 1 \dots K$ **do**

    $\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle)\right]$ // recursion:

    // here, $\phi_{l+1}(\dots k', k \dots)$ is the set of features associated with

    // output position $l+1$ and two adjacent labels k' and k at that position

    $\zeta_{l+1,k} \leftarrow$ the k' that achieves the maximum above // store backpointer

  end for

end for

$\boldsymbol{y} \leftarrow \langle 0, 0, \dots, 0 \rangle$ // initialize predicted output to L-many zeros

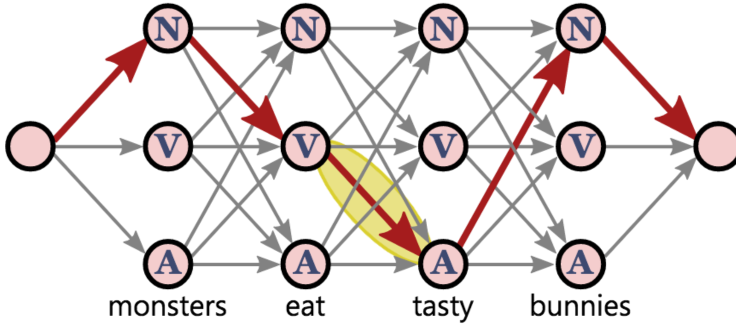
$y_L \leftarrow \operatorname{argmax}_k \alpha_{L,k}$ // extract highest scoring final label

for $l = L\text{-}1 \dots 1$ **do**

  $y_l \leftarrow \zeta_{l, y_{l+1}}$ // traceback ζ based on y_{l+1}

end for

return $\boldsymbol{y}$ // return predicted output



A more general approach for argmax Integer Linear Programming

- ILP: optimization problem of the form, for a fixed vector a

$$\max_{z} \quad \boldsymbol{a} \cdot \boldsymbol{z} \quad \text{subj. to} \quad \text{linear constraints on } \boldsymbol{z}$$

  - With integer constraints

- Pro: can leverage well-engineered solvers (e.g., Gurobi)
- Con: not always most efficient

POS tagging as ILP

- Markov features as binary indicator variables

$$z_{l,k',k} = \mathbf{1}[\text{label } l \text{ is } k \text{ and label } l-1 \text{ is } k']$$

  - Output sequence: y(z) obtained by reading off variables z

- Define a such that a.z is equal to score

$$a_{l,k',k} = \boldsymbol{w} \cdot \phi_l(\boldsymbol{x}, \langle \ldots, k', k \rangle)$$

- Enforcing constraints for well formed solutions

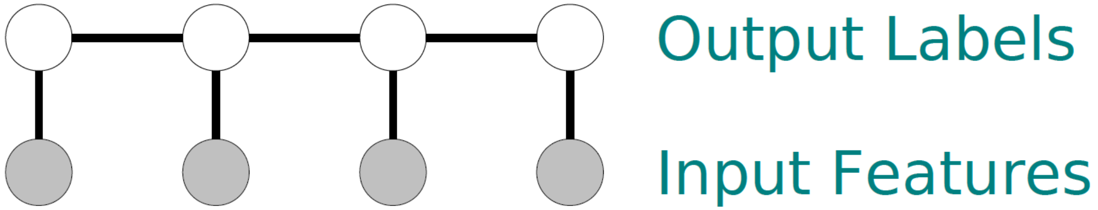
1. That all the zs are binary. That's easy: just say $z_{l,k',k} \in \{0,1\}$, for all l, k', k.
2. That for a given position l, there is exactly one active z. We can do this with an equality constraint: $\sum_k \sum_{k'} z_{l,k',k} = 1$ for all l.
3. That the zs are internally consistent: if the label at position 5 is supposed to be "noun" then both $z_{5,\cdot,\cdot}$ *and* $z_{6,\cdot,\cdot}$ need to agree on this. We can do this as: $\sum_{k'} z_{l,k',k} = \sum_{k''} z_{l+1,k,k''}$ for all l, k. Effectively what this is saying is that $z_{5,?,\text{verb}} = z_{6,\text{verb},?}$ where the "?" means "sum over all possibilities."

Sequence labeling

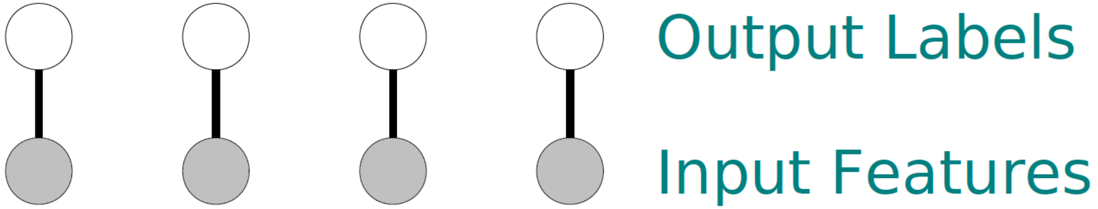
- Structured perceptron
  - A general algorithm for structured prediction problems such as sequence labeling
- The Argmax problem
  - Efficient argmax for sequences with Viterbi algorithm, given some assumptions on feature structure
  - A more general solution: Integer Linear Programming
- Loss-augmented argmax
  - Hamming Loss

How much does this structure matter?

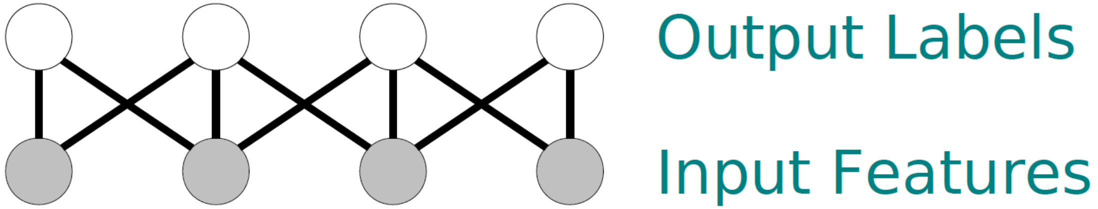
- **Structured models:** accurate but slow



- **Independent models:** less accurate but fast

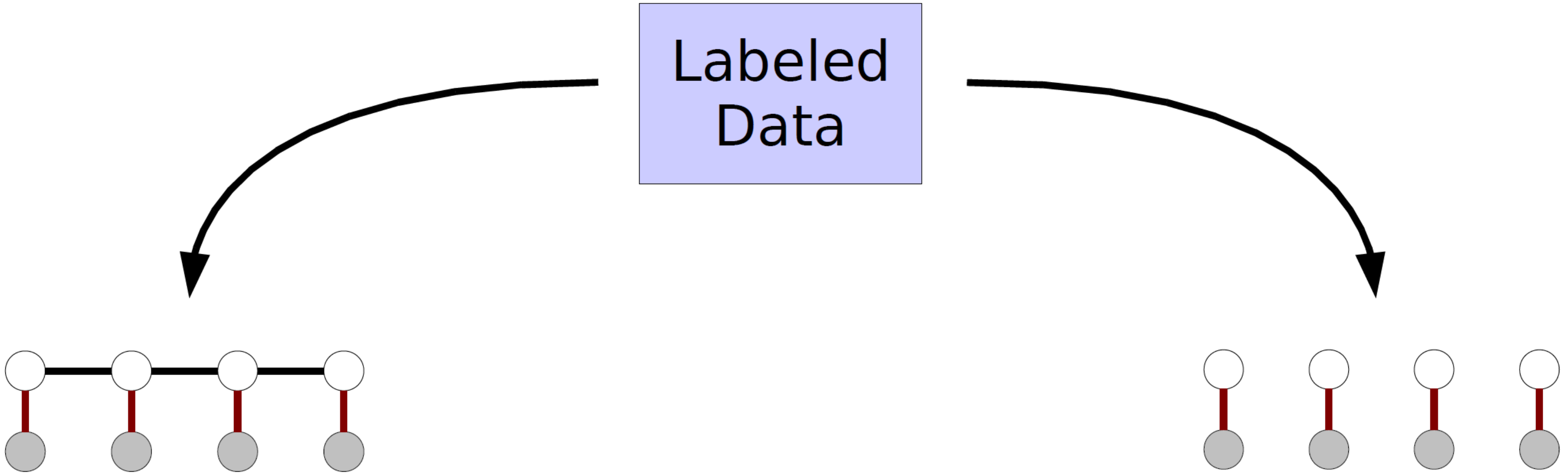


- **Goal:** transfer power to get fast+accurate



- **Questions:** are independent models…
  - … expressive enough? (approximation error)
  - … easy to learn? (estimation error)

“Compiling” structure away

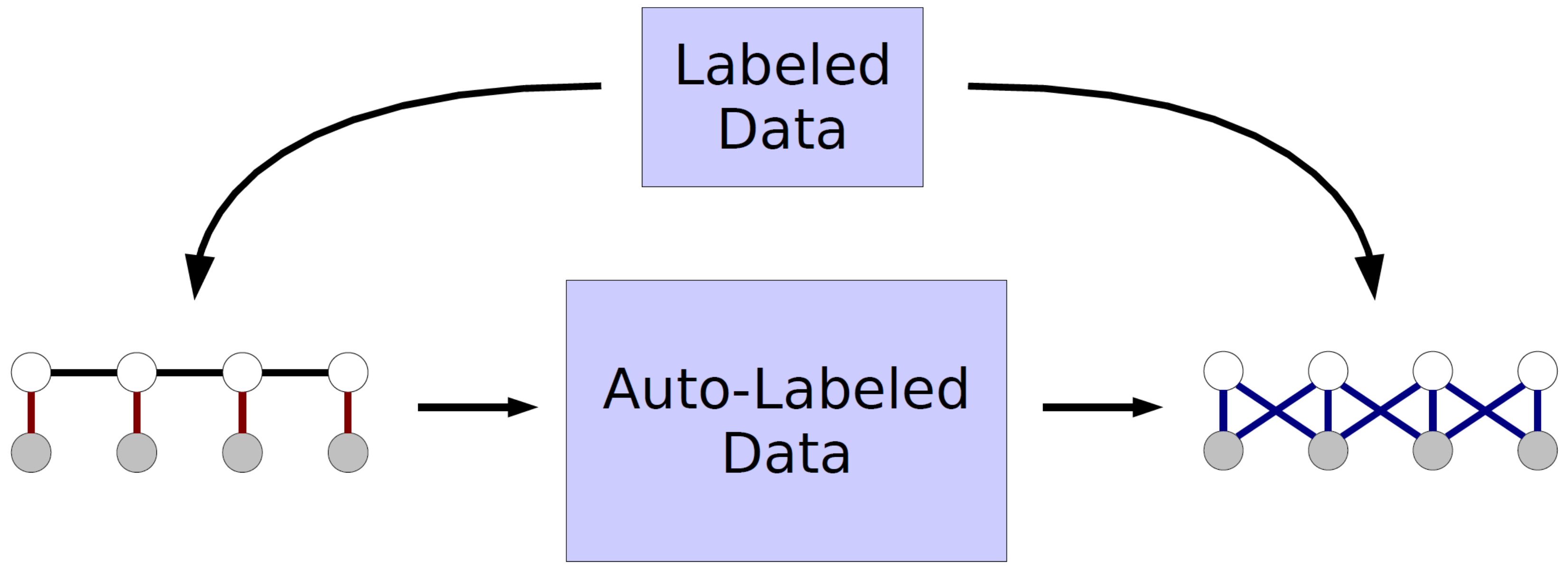


CRF(f_1) POS: 95.0%
NER: 75.3%

IRL(f_1) POS: 91.7%
NER: 69.1%

f_1 = words/prefixes/suffixes/forms

“Compiling” structure away



$\mathrm{CRF}(f_1)$ POS: 95.0% NER: 75.3%

f_1 = words/prefixes/suffixes/forms

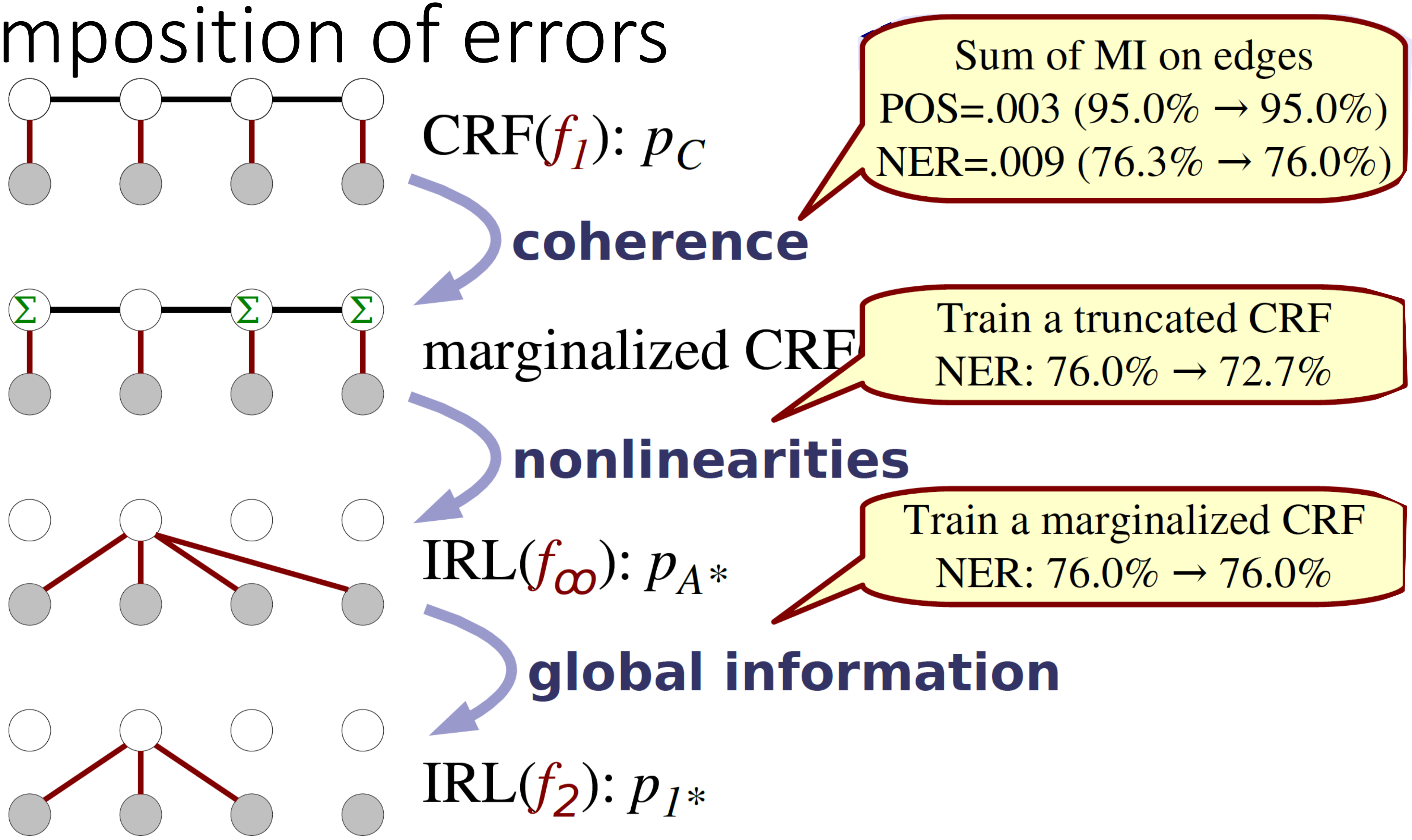
$f_2 = f_1$ applied to a larger window

$\mathrm{IRL}(f_1)$ POS: 91.7% NER: 69.1%

$\mathrm{IRL}(f_2)$ POS: 94.4% NER: 66.2%

$\mathrm{CompIRL}(f_2)$ POS: 95.0% NER: 72.7%

Decomposition of errors

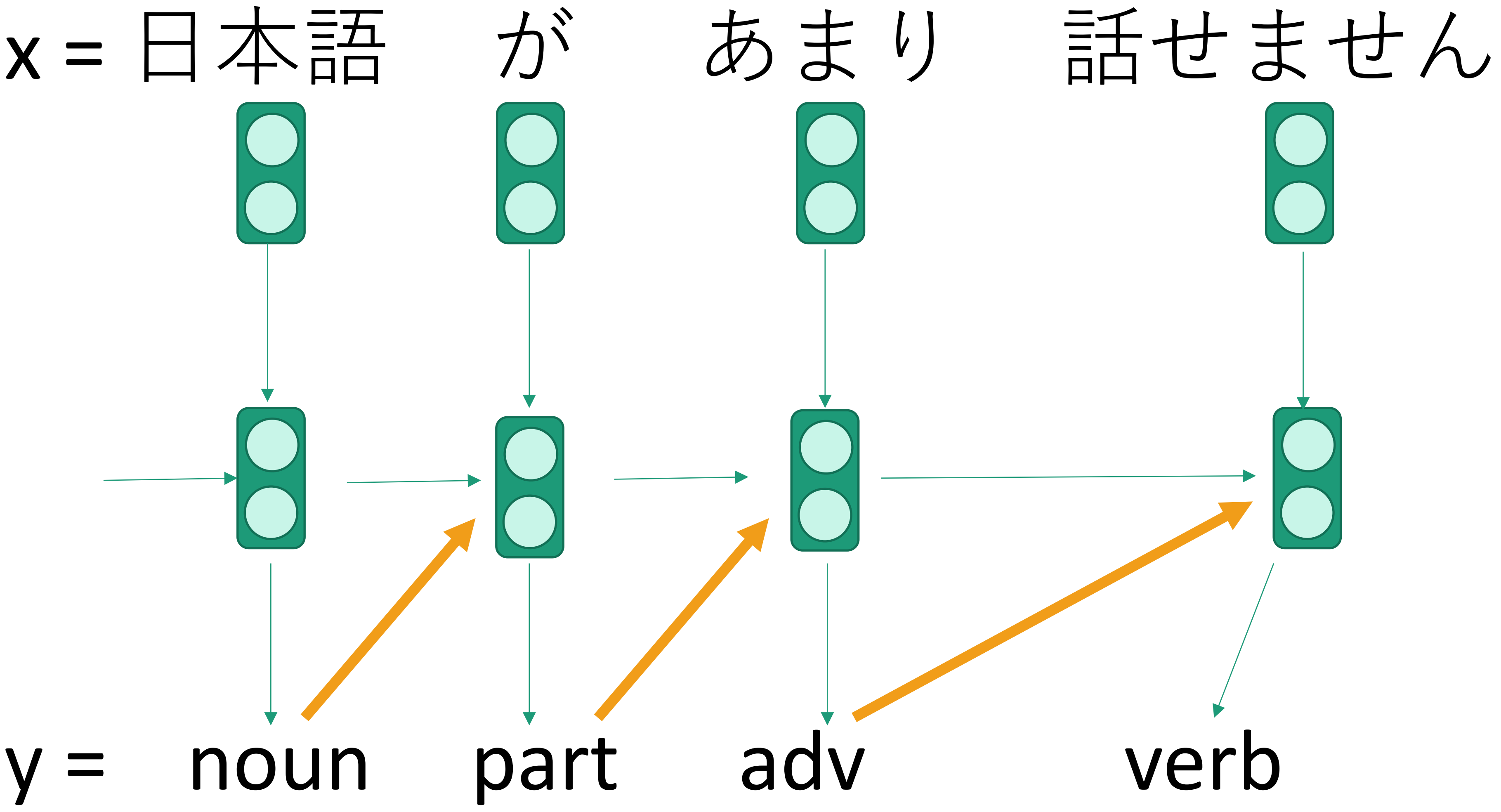


Theorem:

$$\mathrm{KL}(p_C \,||\, p_{1*}) = \mathrm{KL}(p_C \,||\, p_{MC}) + \mathrm{KL}(p_{MC} \,||\, p_{A*}) + \mathrm{KL}(p_{A*} \,||\, p_{1*})$$

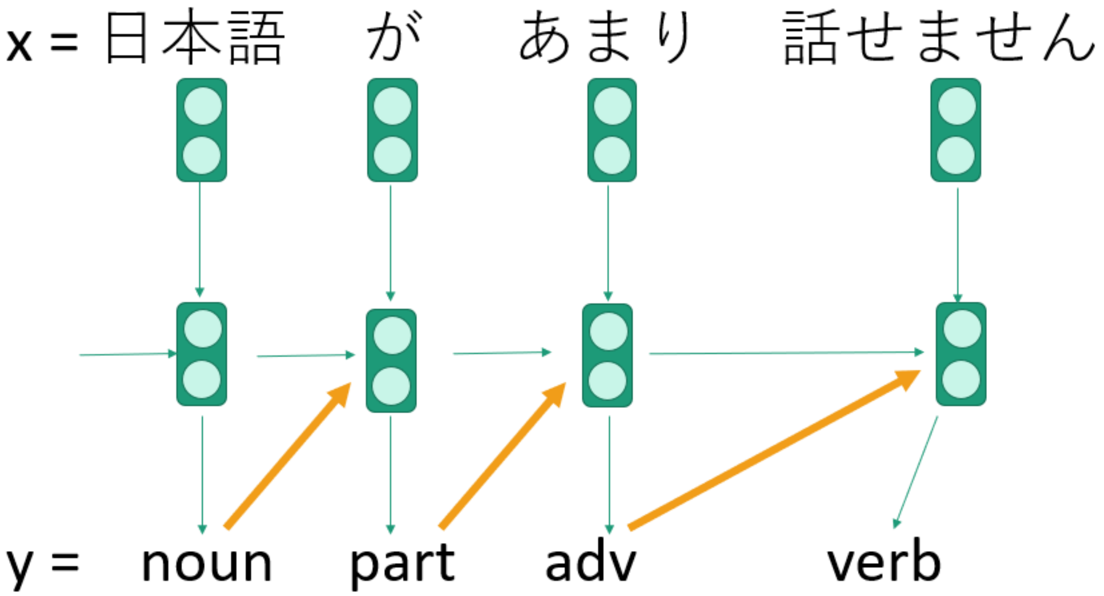
Non-linearities capture most information
… but structure still matters when not enough data

RNN sequence labeling

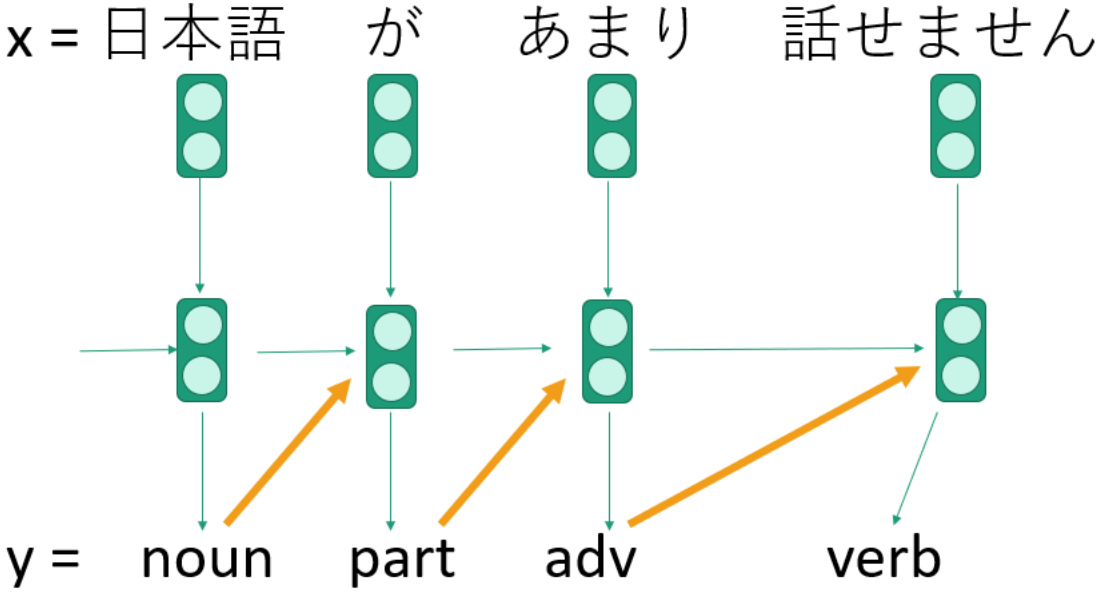


Prediction (aka “decoding”)

- Sampling:
- For n = 1 .. N:
  - y[n] ~ P(Y | X, y[1], …, y[n-1])
- Return y



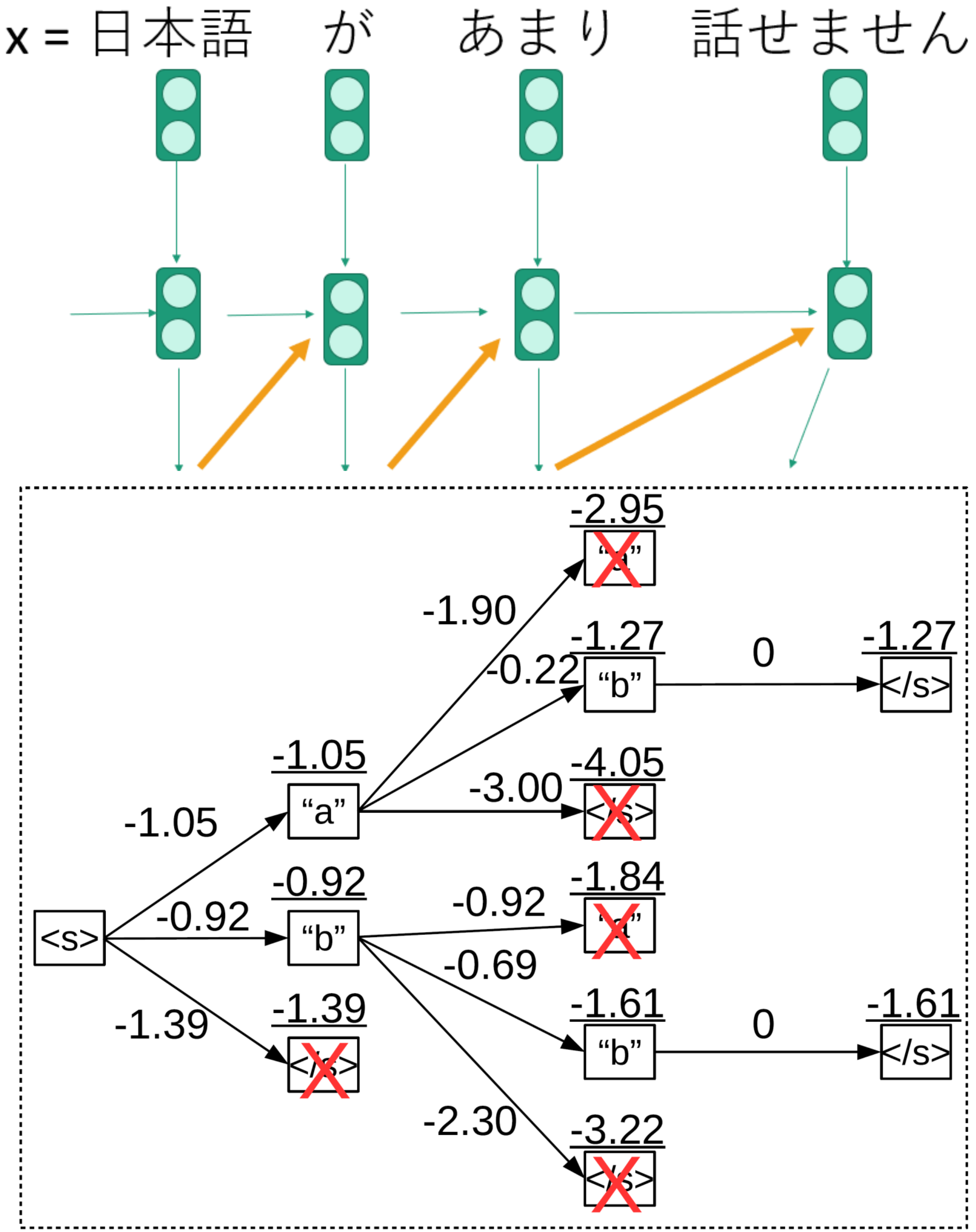
Prediction (aka “decoding”)



- Greedy search:
- For n = 1 .. N:
  - y[n] ← argmax_y P(y | X, y[1], …, y[n-1])
- Return y
- Issue: “label bias problem”

Prediction (aka “decoding”)

- Beam search:
- Consider *b* top hypothesis at each time step
- Example with beam size 2
  - (with log probs on edges)



Today

- Sequence labeling as independent predictions
- Structured perceptron for sequence labeling
- Do we really need structured features?
- Recurrent neural network taggers