Machine Learning 2

DS 4420 - Spring 2020

Structured prediction, II

Byron C Wallace

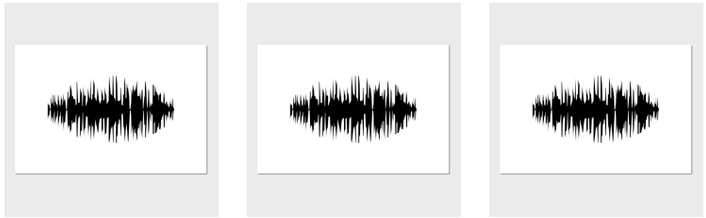
Today

- From HMMs to MEMMs to CRFs

Structured output spaces

y_1 y_2 y_3

"*Play Kanye West*"



x_1 x_2 x_3

Structured output spaces

Source: http://cocodataset.org/

Space of problems

Given	Predict	Type?
An image	Contains a cat?	**Classification**
An image	Coordinates that outline all cats	**Structured prediction**
A tweet	Names in the tweet	**Structured prediction**
A tweet	Sentiment in tweet	**Classification**

A generative model of sequences

$$P(X_1 = x_1 \ldots X_n = x_n, Y_1 = y_1 \ldots Y_n = y_n)$$

A generative model of sequences

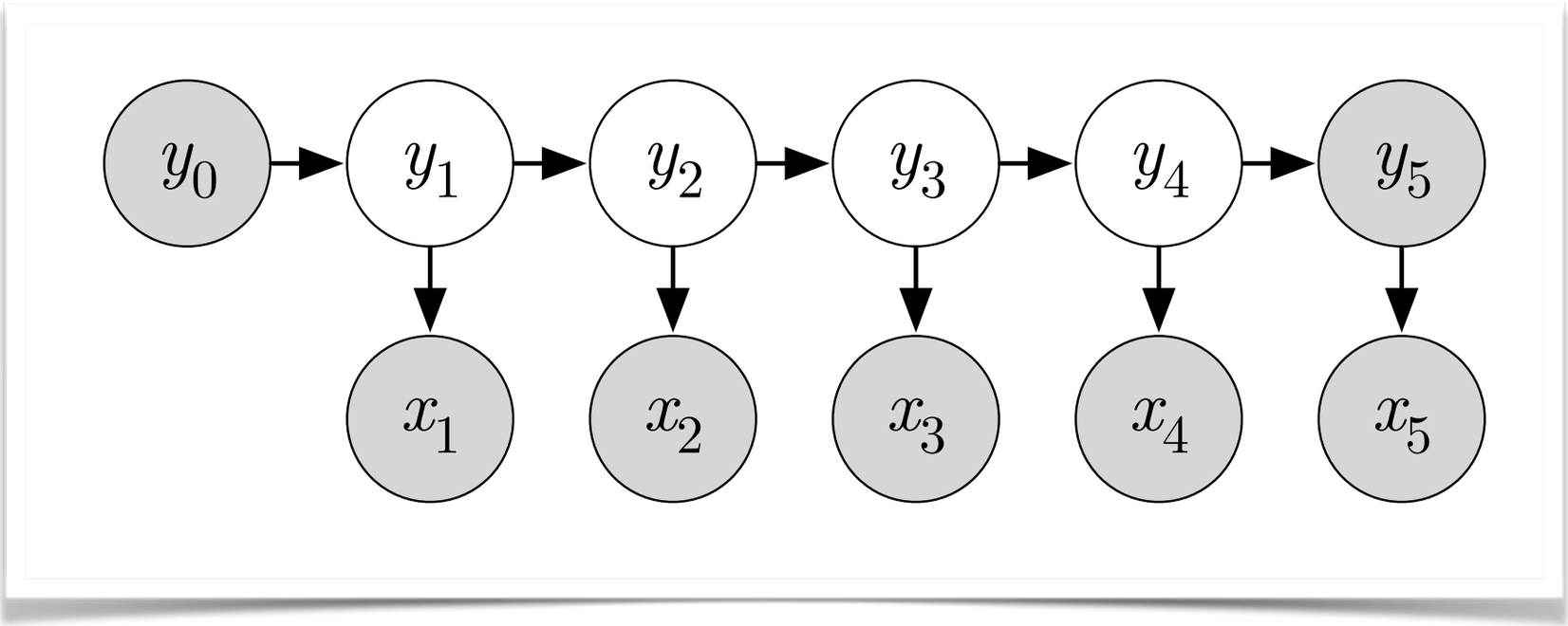
$$P(X_1 = x_1 \dots X_n = x_n, Y_1 = y_1 \dots Y_n = y_n)$$

$$= \prod_{i=1}^{n+1} P(y_i|y_{i-1}) \prod_{i=1}^{n} P(x_i|y_i)$$

Transition probability

Emission probability

Graphical Model (HMMs)



Limitations to HMMs

- We are restricted to features that have a coherent "generative story"
- Why bother "modeling" x — it's given! What we really care about is $p(y|x)$

Generative v discriminative

Generative

Model **joint** distribution P(x,y)
Can **generate** new “examples”
To *predict* y, use Bayes’ rule

Generative v discriminative

Generative

Model **joint** distribution P(x,y)
Can **generate** new “examples”
To *predict* y, use Bayes’ rule

Discriminative

Model **conditional** distribution P(y|x)
Not as amenable to semi-supervised settings; cannot readily “generate” new samples

Enter Max Entropy Markov Models (MEMMs)

- These extend standard *log-linear* models to capture structure in the outputs.

- A bit like the *structured perceptron* we introduced last time, but explicitly model conditional probabilities of labels.

Log-Linear Models

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

Log-Linear Models

measures *plausibility* of *y* given *x*

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

Log-likelihood

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

Log-likelihood

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

$$LL(w) = \sum_i \log\, p(y_i|x_i,w)$$

Back to sequences

Want to model conditional probability of sequence of *y*

$$p(y_1, y_2, ...y_m|x_1, x_2, ..., x_m)$$

Back to sequences

Want to model conditional probability of sequence of *y*

$$p(y_1, y_2, ...y_m|x_1, x_2, ..., x_m)$$

i here used to denote "index"

$$\prod_{i}^{m} p(y_i|y_1, ..., y_{i-1}, x_1, ...x_m)$$

Back to sequences

Want to model conditional probability of sequence of *y*

$$p(y_1, y_2, ...y_m | x_1, x_2, ..., x_m)$$

i here used to denote "index"

$$\prod_i^m p(y_i | y_1, ..., y_{i-1}, x_1, ...x_m)$$

$$= \prod_i^m p(y_i | y_{i-1}, x_1, ...x_m)$$

MEMMs

Combine

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

MEMMs

Combine

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

With

$$p(y_1, y_2, ...y_m | x_1, x_2, ..., x_m) = \prod_i^m p(y_i | y_{i-1}, x_1, ...x_m)$$

MEMMs

Combine

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

With

$$p(y_1, y_2, ...y_m | x_1, x_2, ..., x_m) = \prod_i^m p(y_i | y_{i-1}, x_1, ...x_m)$$

For:

$$p(y|y_{i-1}, x_1, ..., x_m, w) = \quad ???$$

MEMMs

Combine

$$p(y|x,w) = \frac{\exp(w \cdot \phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x,y'))}$$

With

$$p(y_1, y_2, ...y_m | x_1, x_2, ..., x_m) = \prod_i^m p(y_i | y_{i-1}, x_1, ...x_m)$$

For:

$$p(y|y_{i-1}, x_1, ..., x_m, w) = \frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

Training

$$p(y|y_{i-1}, x_1, ..., x_m, w) = \frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

What will our training examples look like?

Training

$$p(y|y_{i-1}, x_1, ..., x_m, w) = \frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

What will our training examples look like?

$$\{y, [y_{i-1}, x_1, ..., x_m]\}$$

Predicting

$$\arg\max_{\hat{y} \in \mathcal{Y}} p(y|x, w)$$

Predicting

$$\arg\max_{\hat{y} \in \mathcal{Y}} p(y|x, w)$$

Viterbi (see last lecture)!

HMMs v MEMMs

HMM $p(y_i|y_{i-1})p(x_i|y_i)$

MEMM $\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$

ϕ permits richer representations!

Feature engineering

MEMM $$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

ϕ permits richer representations!

Consider NER:

Yesterday secretary of state **Mike Pompeo** meet with Ethiopia's Prime Minister **Abiy Ahmed**

What are some potential features here?

Feature engineering

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

Suppose we have some *deep neural network* that yields embeddings for each word; we could stack a MEMM on top of this.

What would reasonable features be here?

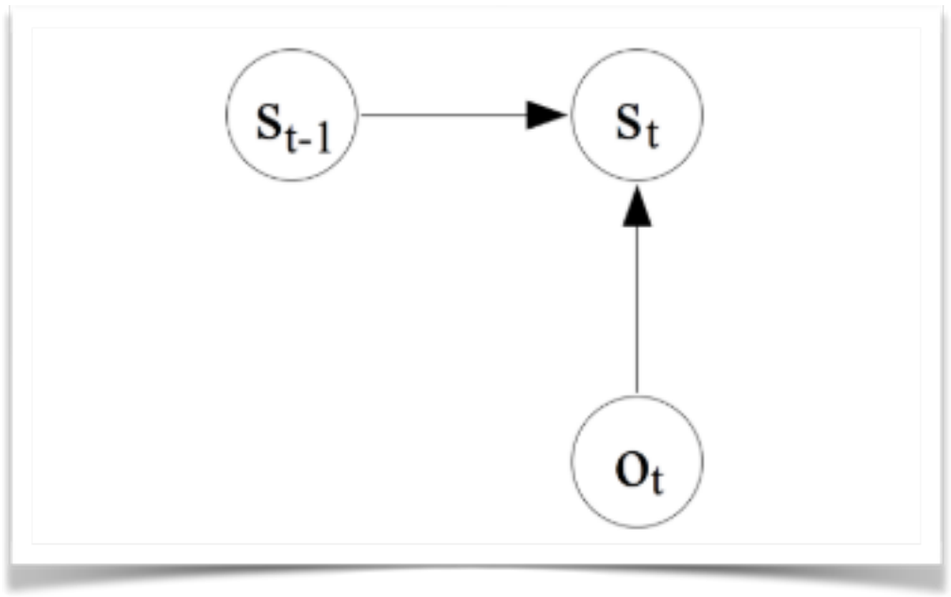
The “label bias” problem

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

The “label bias” problem

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

Locally re-normalized; states are *competing* against each other



McCallum et al., 2001

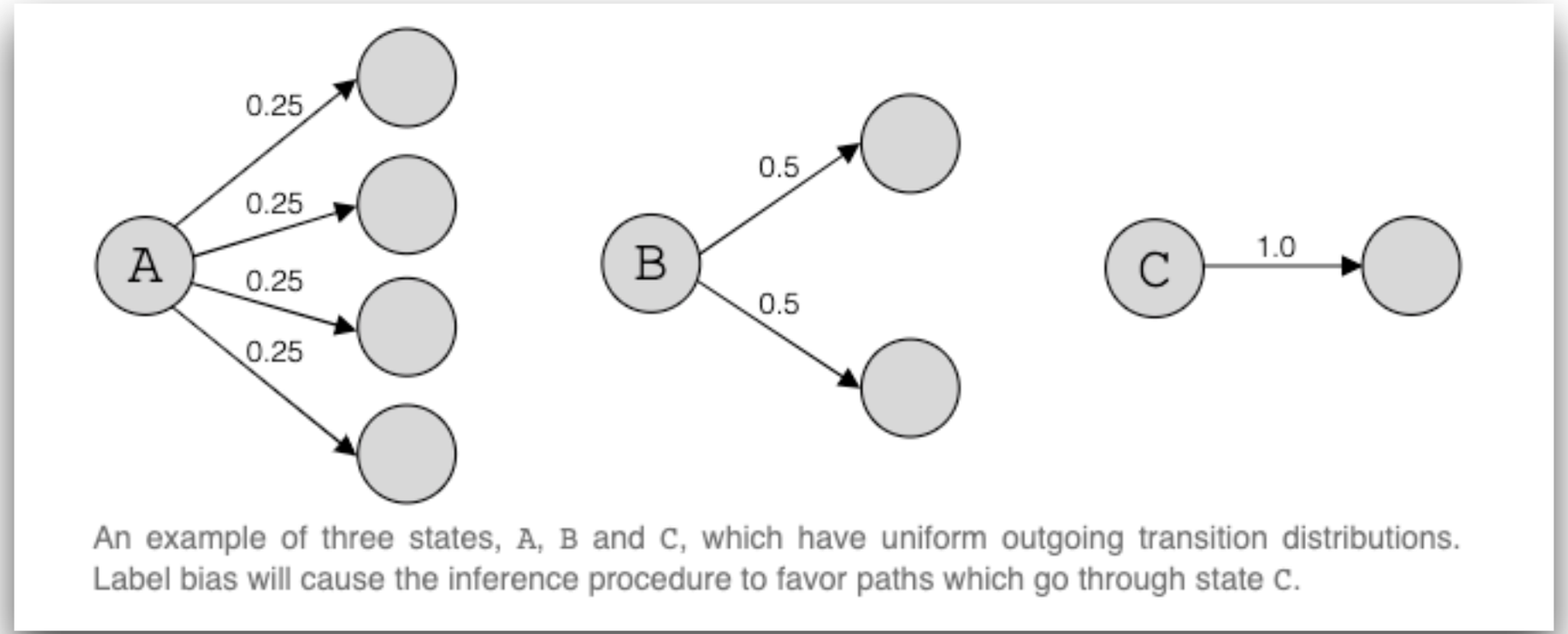
The “label bias” problem

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

Locally re-normalized; states are *competing* against each other

In an extreme case, a particular *y* may *always* place all mass on some other *y’* — ignoring the local input!

The “label bias” problem



An example of three states, A, B and C, which have uniform outgoing transition distributions. Label bias will cause the inference procedure to favor paths which go through state C.

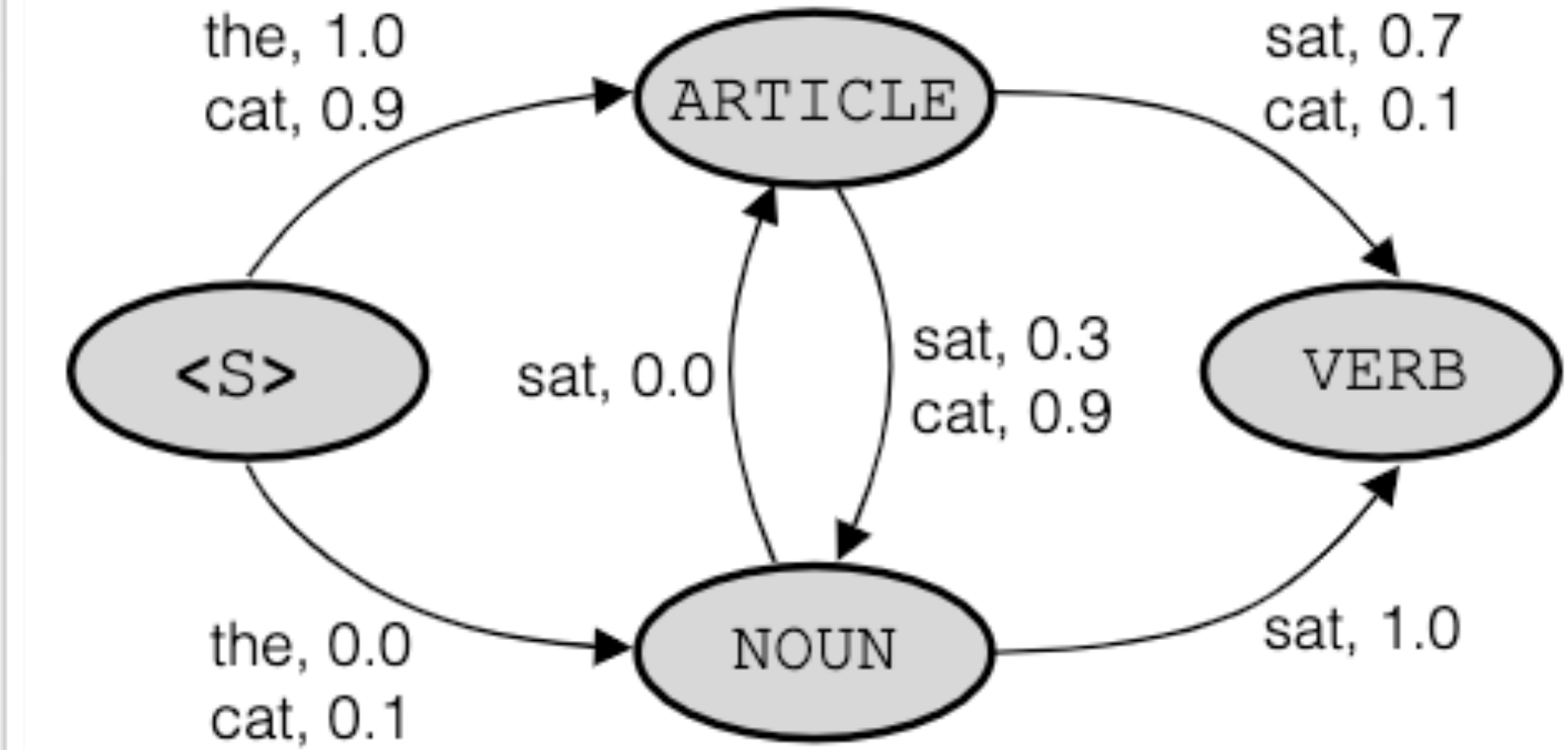
Figure from Awni Hannun, https://awni.github.io/label-bias/

Example

y [ARTICLE, NOUN, VERB]

V [the, cat, sat]

Example from Awni Hannun, https://awni.github.io/label-bias/

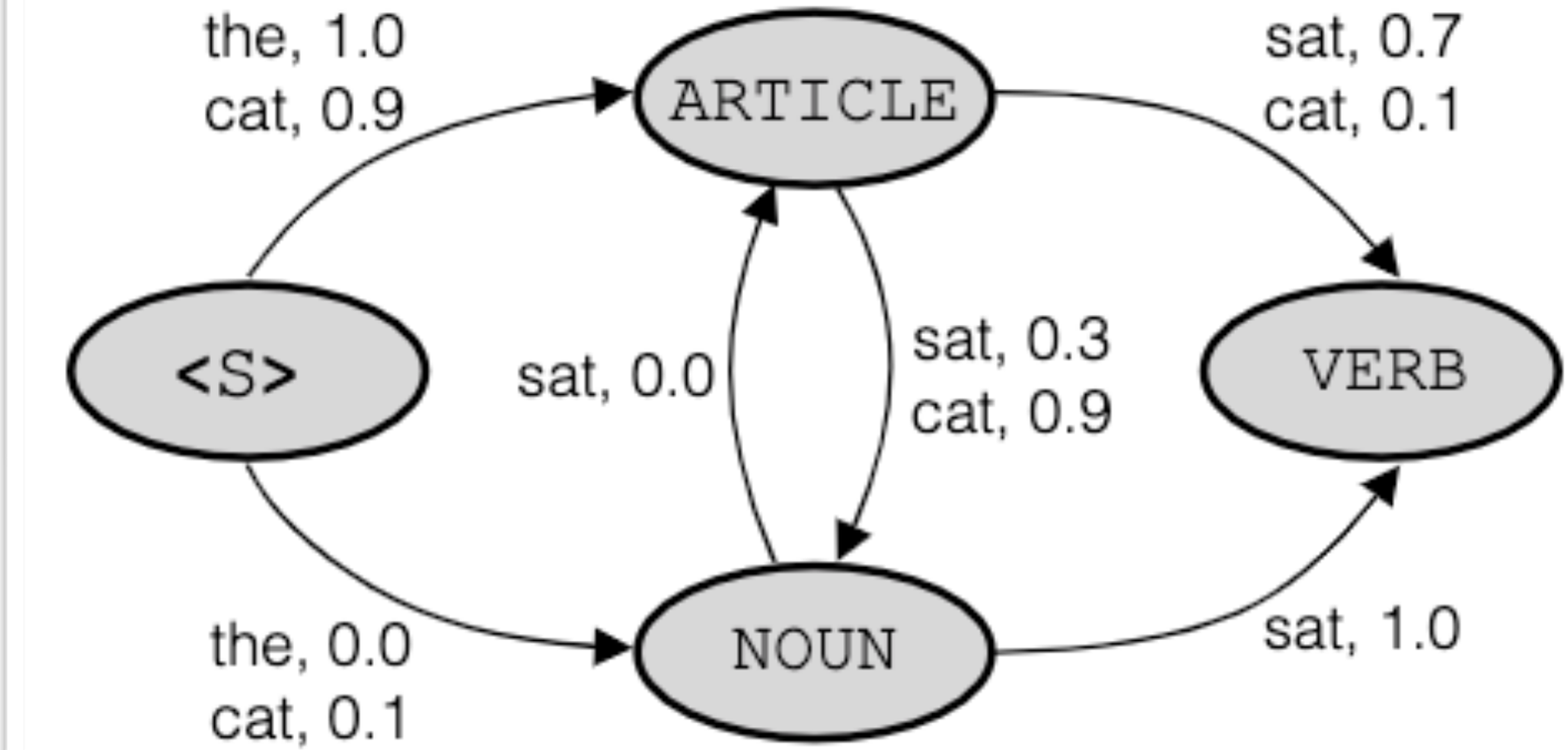


V [cat, sat, the]

x: *cat sat*

y: N V

Example from Awni Hannun, https://awni.github.io/label-bias/



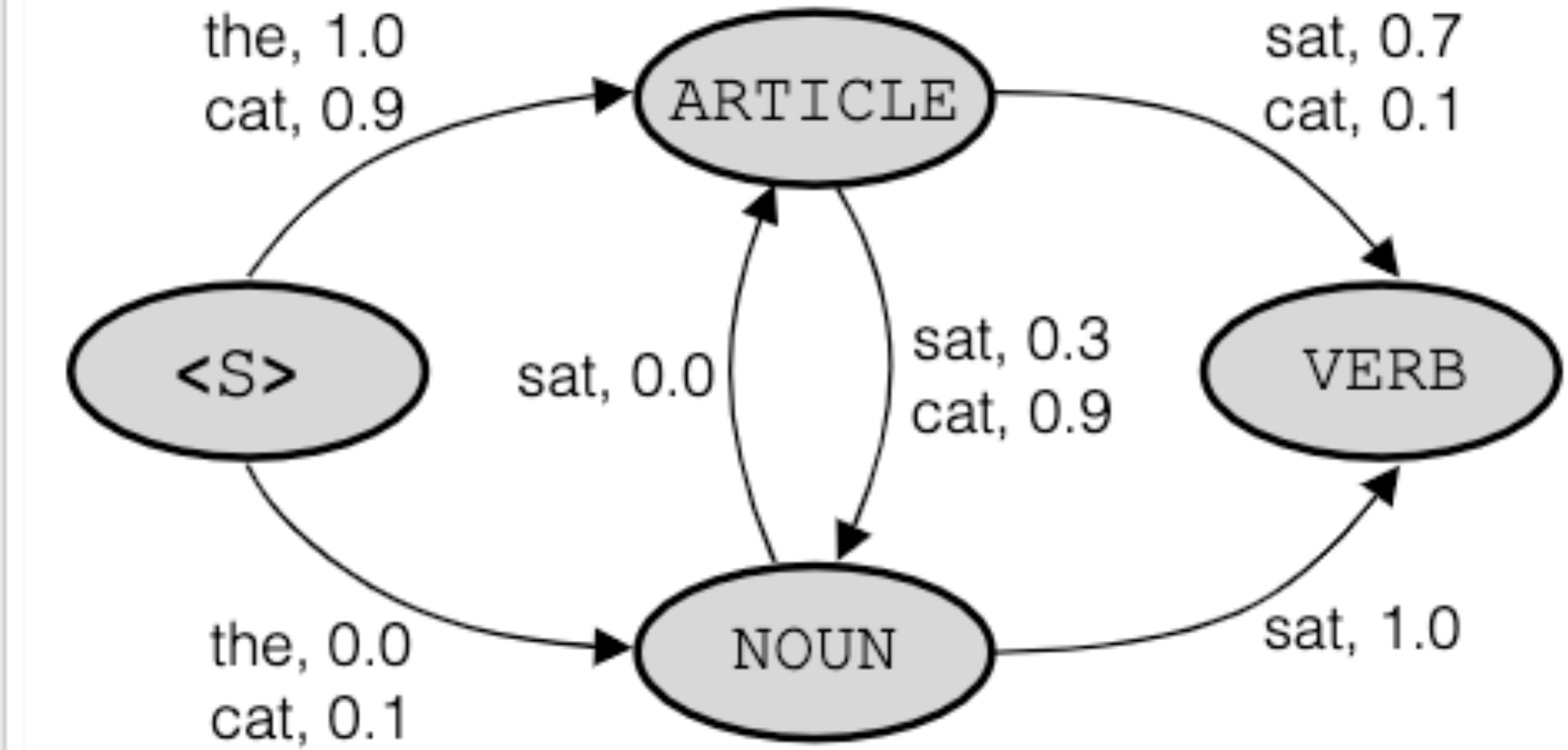
V [cat, sat, the]

x: *cat sat*

y: N V

$$p(\mathrm{N}, \mathrm{V} | \mathrm{cat}, \mathrm{sat}) = \; ???$$

Example from Awni Hannun, https://awni.github.io/label-bias/



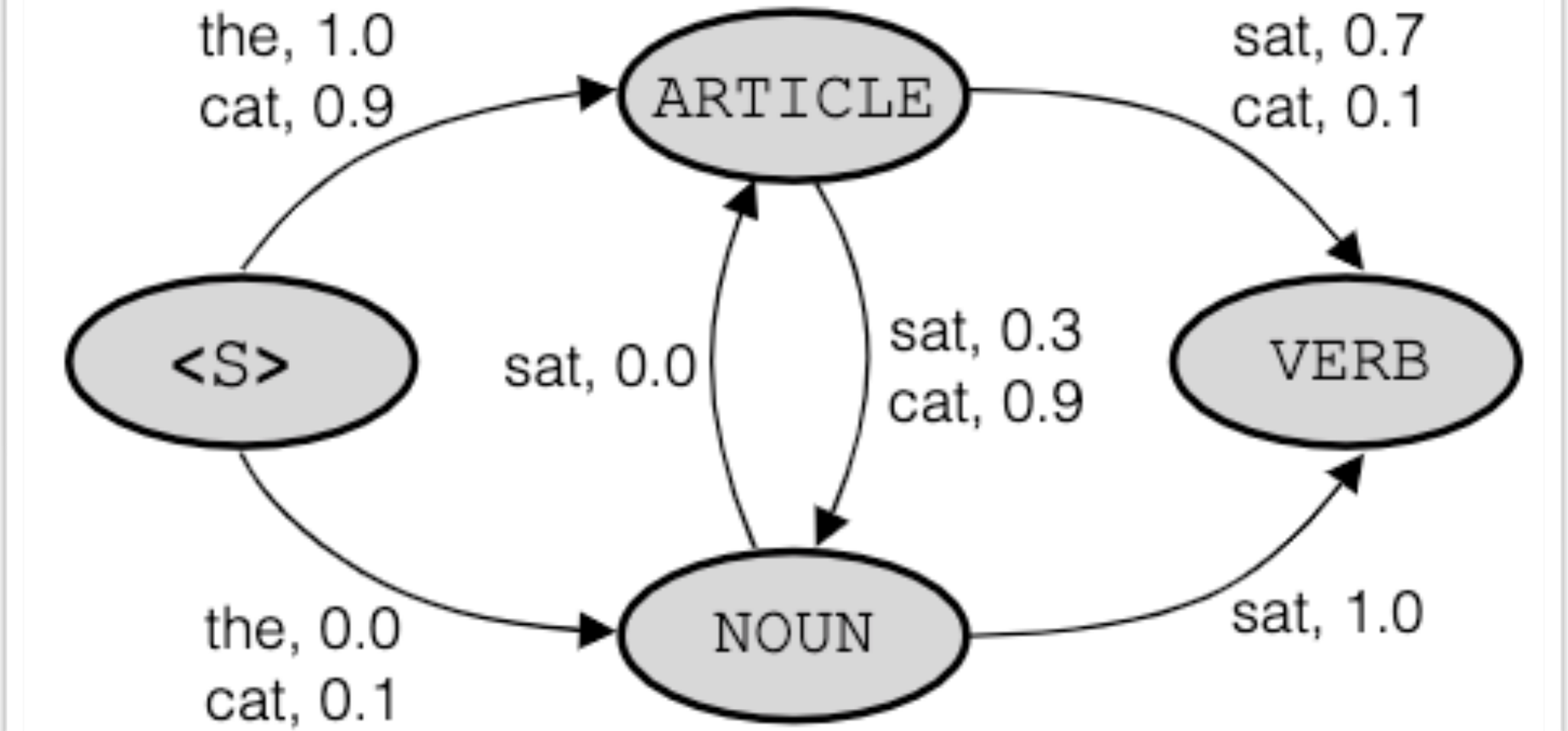
V [cat, sat, the]

x: *cat sat*

y: N V

$$p(\mathrm{N}, \mathrm{V}|\mathrm{cat}, \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

Example from Awni Hannun, https://awni.github.io/label-bias/



V [cat, sat, the]

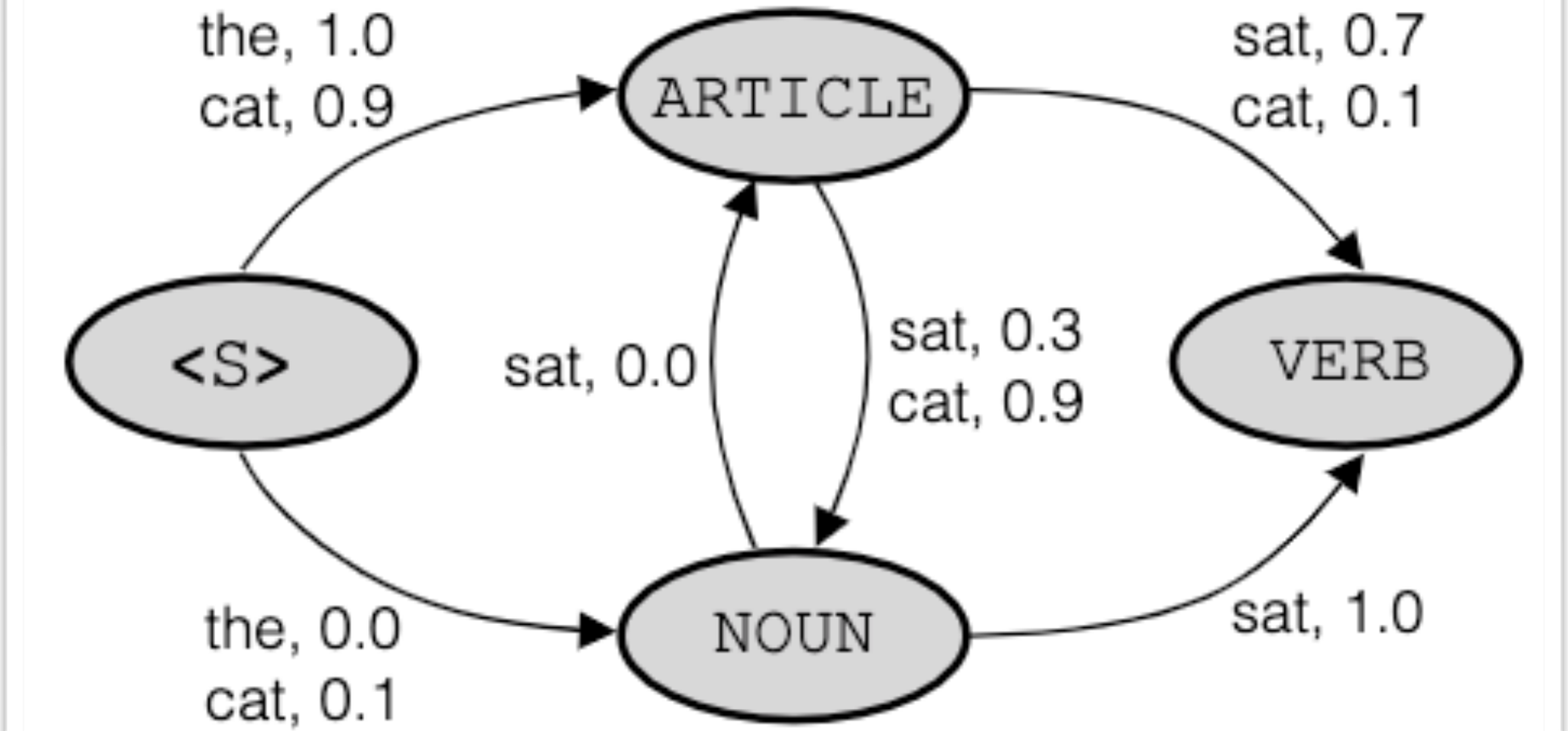
x: *cat sat*

y: N V

$$p(\mathrm{N},\ \mathrm{V}|\mathrm{cat},\ \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

$$p(\mathrm{A},\ \mathrm{N}|\mathrm{cat},\ \mathrm{sat}) = \quad ???$$

Example from Awni Hannun, https://awni.github.io/label-bias/



V [cat, sat, the]

x: *cat sat*

y: N V

$$p(\mathrm{N},\ \mathrm{V}|\mathrm{cat},\ \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

$$p(\mathrm{A},\ \mathrm{N}|\mathrm{cat},\ \mathrm{sat}) = 0.9 \cdot 0.3 = 0.27$$

Example from Awni Hannun, https://awni.github.io/label-bias/

Why is this happening?

$$p(\mathrm{N},\ \mathrm{V}|\mathrm{cat},\ \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

$$p(\mathrm{A},\ \mathrm{N}|\mathrm{cat},\ \mathrm{sat}) = 0.9 \cdot 0.3 = 0.27$$

Example from Awni Hannun, https://awni.github.io/label-bias/

Why is this happening?

$$p(\mathrm{N},\ \mathrm{V}|\mathrm{cat},\ \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

$$p(\mathrm{A},\ \mathrm{N}|\mathrm{cat},\ \mathrm{sat}) = 0.9 \cdot 0.3 = 0.27$$

"cat" rarely seen as first word; poorly calibrated.
But the mass has to go somewhere! Why?

Example from Awni Hannun, https://awni.github.io/label-bias/

Why is this happening?

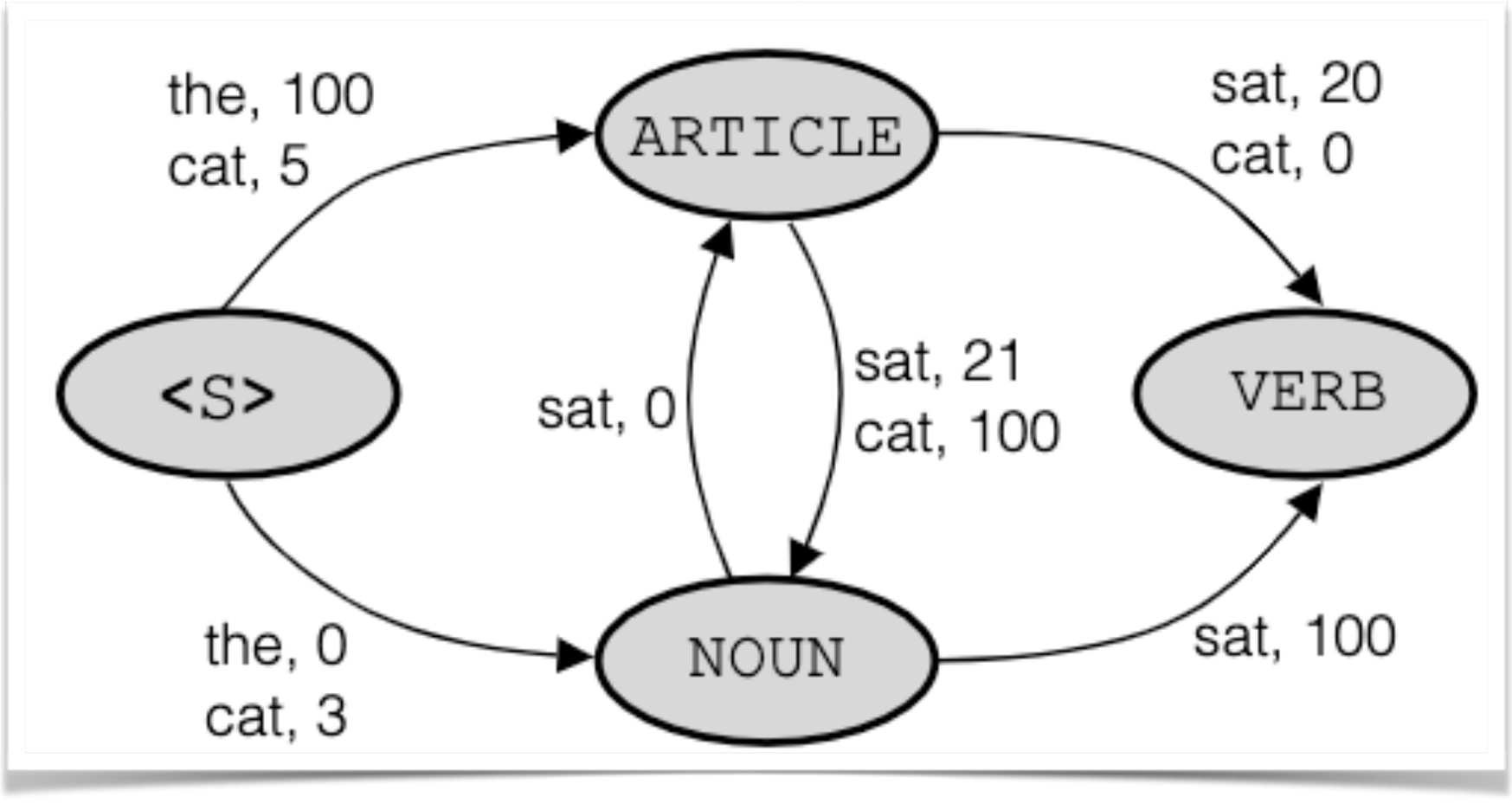
$$p(\mathrm{N},\ \mathrm{V}|\mathrm{cat},\ \mathrm{sat}) = 0.1 \cdot 1.0 = 0.1$$

$$p(\mathrm{A},\ \mathrm{N}|\mathrm{cat},\ \mathrm{sat}) = 0.9 \cdot 0.3 = 0.27$$

"cat" rarely seen as first word; poorly calibrated.
But the mass has to go somewhere! Why?
Transitions are locally normalized

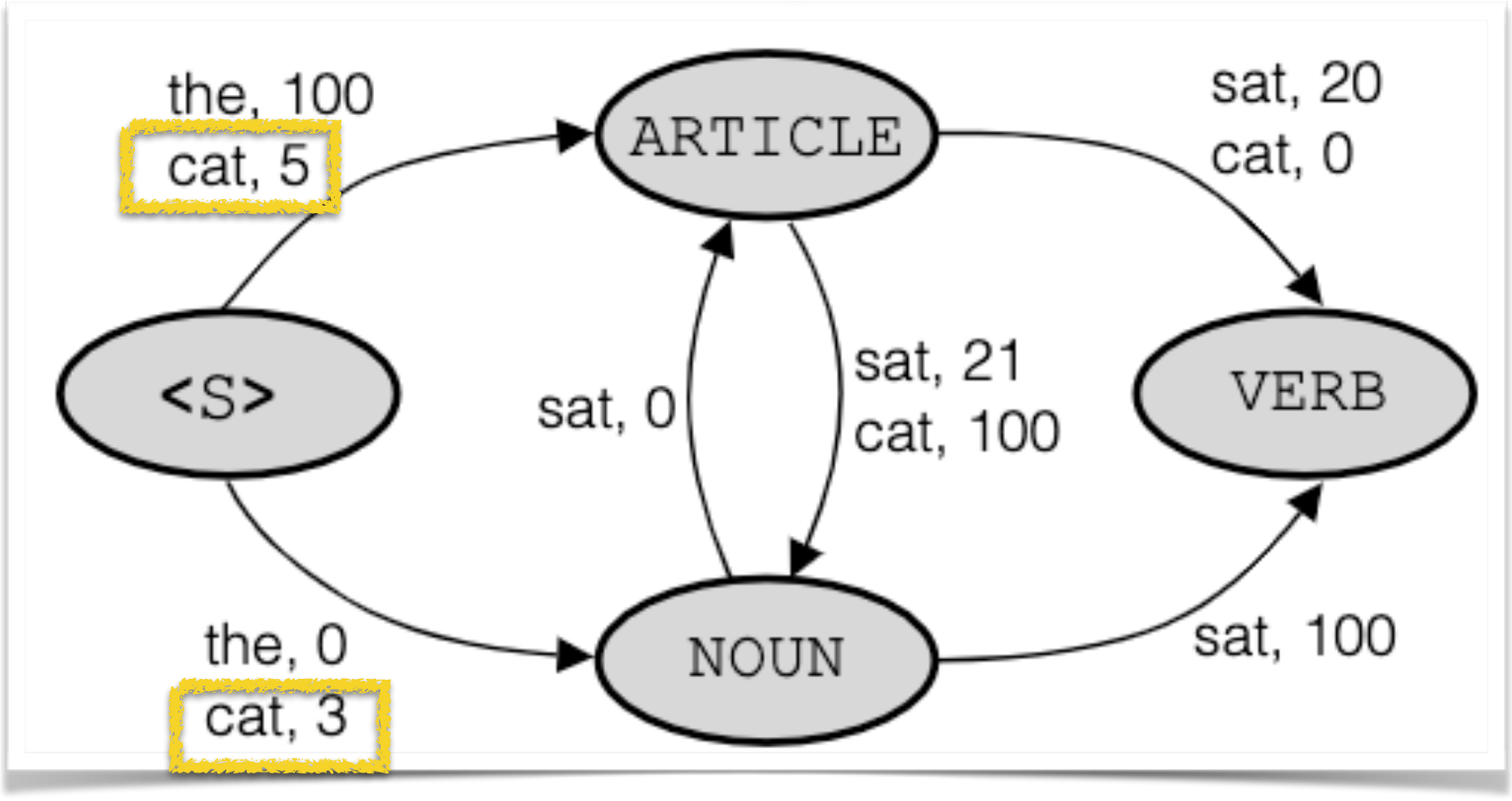
Example from Awni Hannun, https://awni.github.io/label-bias/

Hypothetical **unnormalized** edge scores



Example from Awni Hannun, https://awni.github.io/label-bias/

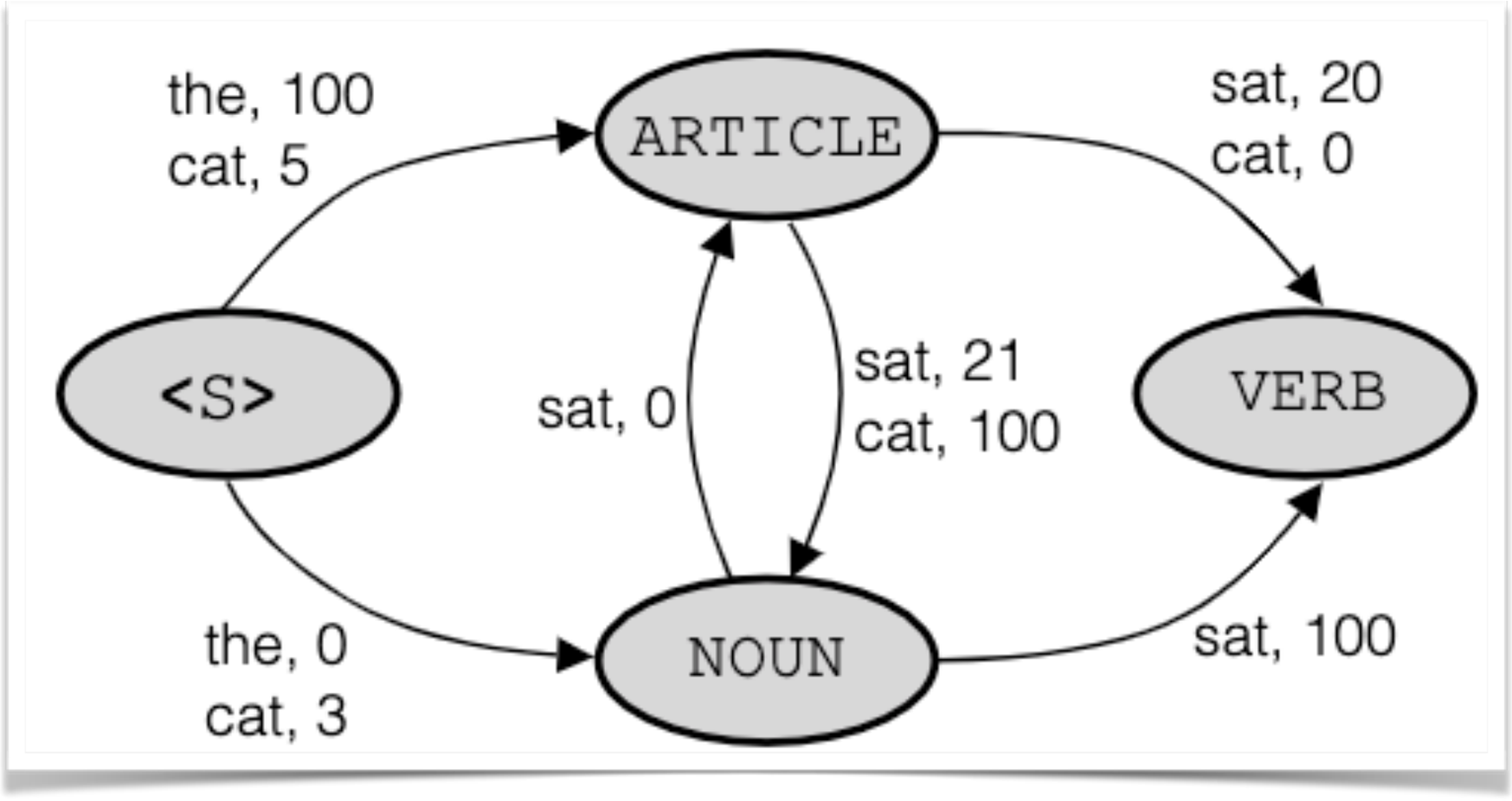
Hypothetical **unnormalized** edge scores



Both are low! We are not confident about what to do with *cat*

Example from Awni Hannun, https://awni.github.io/label-bias/

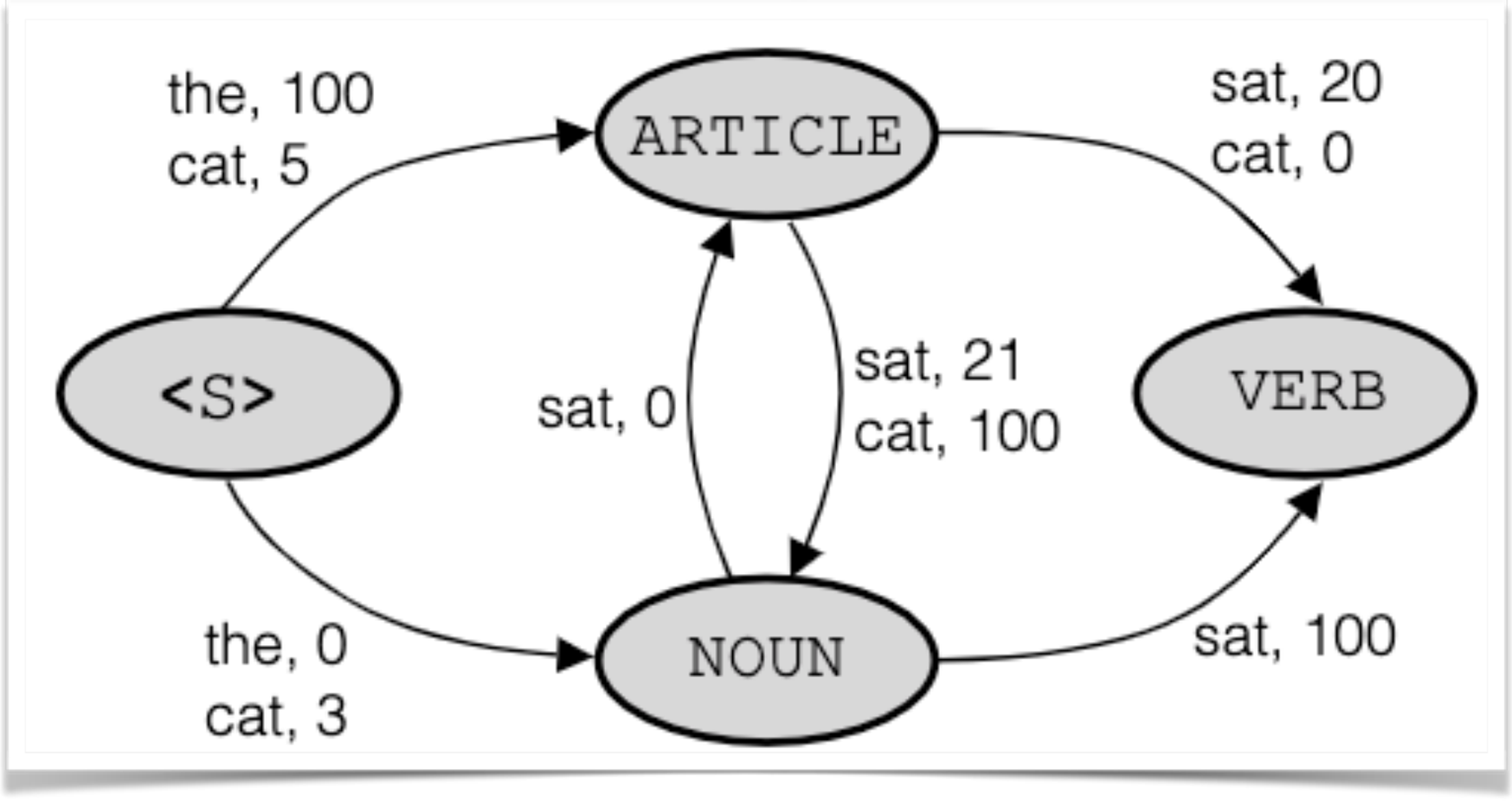
Hypothetical **unnormalized** edge scores



$$\text{score}(\text{A, N}|\text{cat, sat}) = 5 + 21 = 26$$
$$\text{score}(\text{N, V}|\text{cat, sat}) = 3 + 100 = 103$$

Example from Awni Hannun, https://awni.github.io/label-bias/

Hypothetical **unnormalized** edge scores

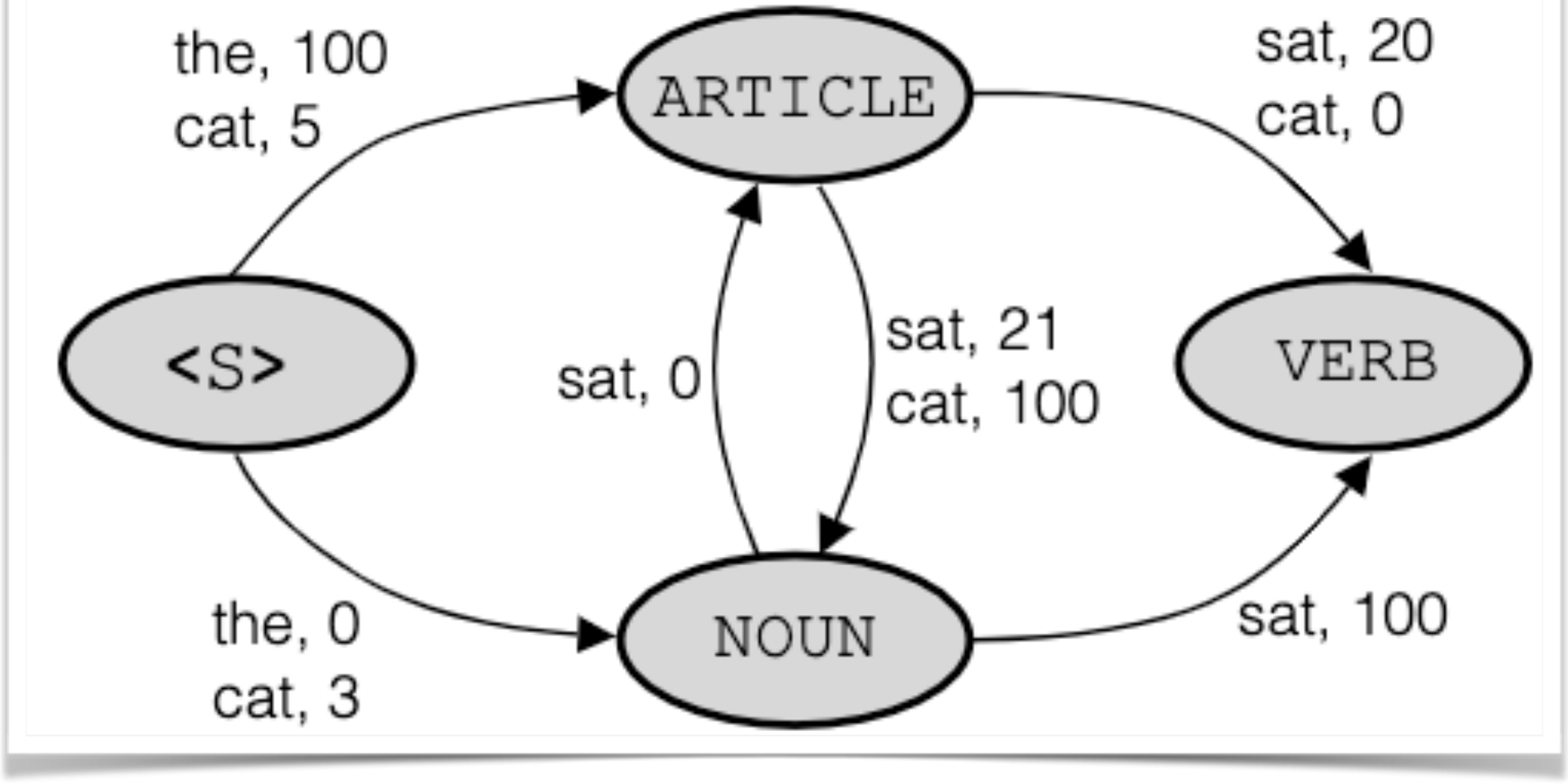


$$\text{score}(\text{A, N}|\text{cat, sat}) = 5 + 21 = 26$$
$$\text{score}(\text{N, V}|\text{cat, sat}) = 3 + 100 = 103$$

Reversed!

Example from Awni Hannun, https://awni.github.io/label-bias/

Hypothetical **unnormalized** edge scores



$$\text{score}(\text{A, N}|\text{cat, sat}) = 5 + 21 = 26$$
$$\text{score}(\text{N, V}|\text{cat, sat}) = 3 + 100 = 103$$

Reversed!

Note: Why add instead of multiply here?

Example from Awni Hannun, https://awni.github.io/label-bias/

Label bias

- Because transitions are *locally* normalized, MEMMs prefer low entropy states
- Difficult to “recover” from mistakes

Example from Awni Hannun, https://awni.github.io/label-bias/

Global scores

$$s(x, y) = \sum_i s(y_i, x_i, y_{i-1})$$

Global scores

$$s(x, y) = \sum_i s(y_i, x_i, y_{i-1})$$

Why can’t we just maximize this, period?

Global scores

$$s(x, y) = \sum_i s(y_i, x_i, y_{i-1})$$

Why can’t we just maximize this, period?
No competition between different labels!

Global normalization

$$p(y|x) = \frac{\exp\{\sum_i s(y_i, x_i, y_{i-1})\}}{\sum_{y'} \exp\{\sum_i s(y'_i, x_i, y'_{i-1})\}}$$

Global normalization

$$p(y|x) = \frac{\exp\{\sum_i s(y_i, x_i, y_{i-1})\}}{\sum_{y'} \exp\{\sum_i s(y'_i, x_i, y'_{i-1})\}}$$

This is a *linear-chain* Conditional Random Field (CRF)

MEMMs vs CRFs

MEMMs *locally* normalize, chain together transition probabilities:

$$p(y|x) = \prod_{i}^{m} p(y_i|y_{i-1}, x_1, ...x_m)$$

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

MEMMs vs CRFs

MEMMs *locally* normalize, chain together transition probabilities:

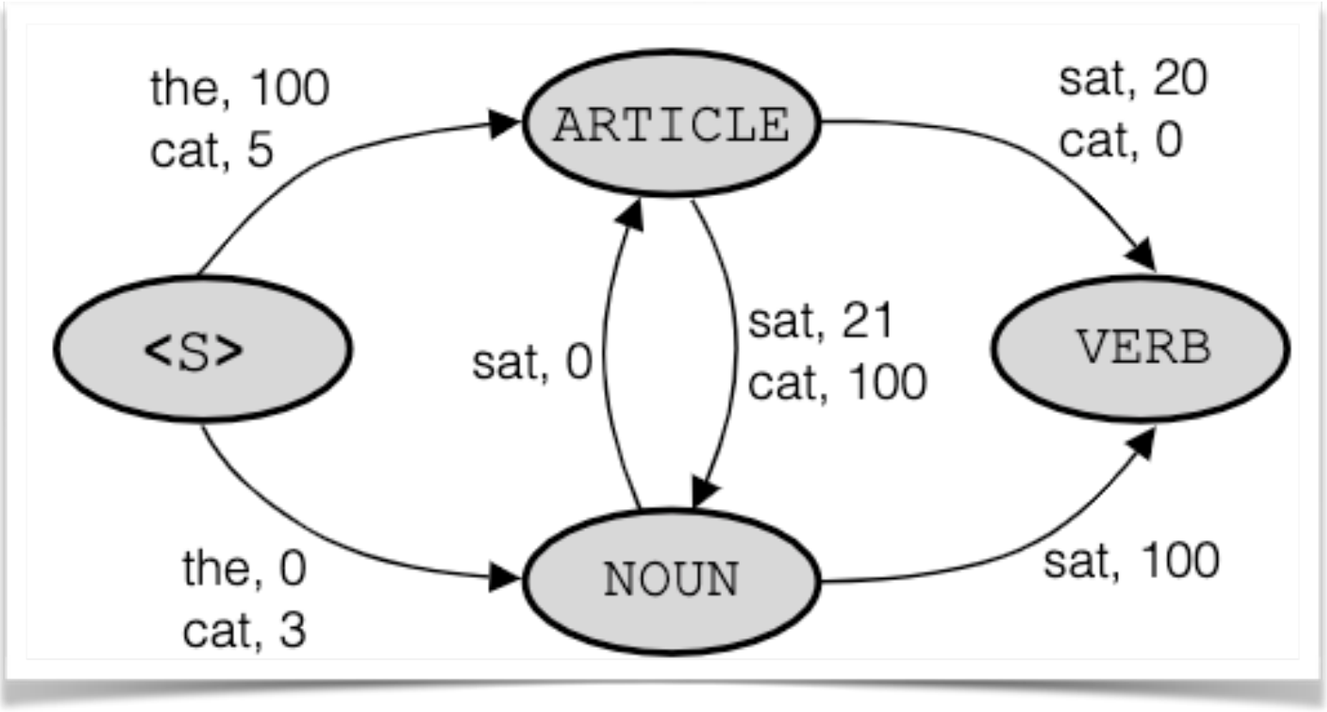
$$p(y|x) = \prod_{i}^{m} p(y_i|y_{i-1}, x_1, ...x_m)$$

$$\frac{\exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y_i))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x_1, ..., x_m, y_{i-1}, y'))}$$

CRFs *globally* normalize

$$p(y|x) = \frac{\exp\{\sum_i s(y_i, x_i, y_{i-1})\}}{\sum_{y'} \exp\{\sum_i s(y'_i, x_i, y'_{i-1})\}}$$

For simplicity assume no self-loops



$$p(y|x) = \frac{\exp\{\sum_i s(y_i, x_i, y_{i-1})\}}{\sum_{y'} \exp\{\sum_i s(y'_i, x_i, y'_{i-1})\}}$$

$\mathcal{Y}$ `[ARTICLE, NOUN, VERB]` V [cat, sat, the]

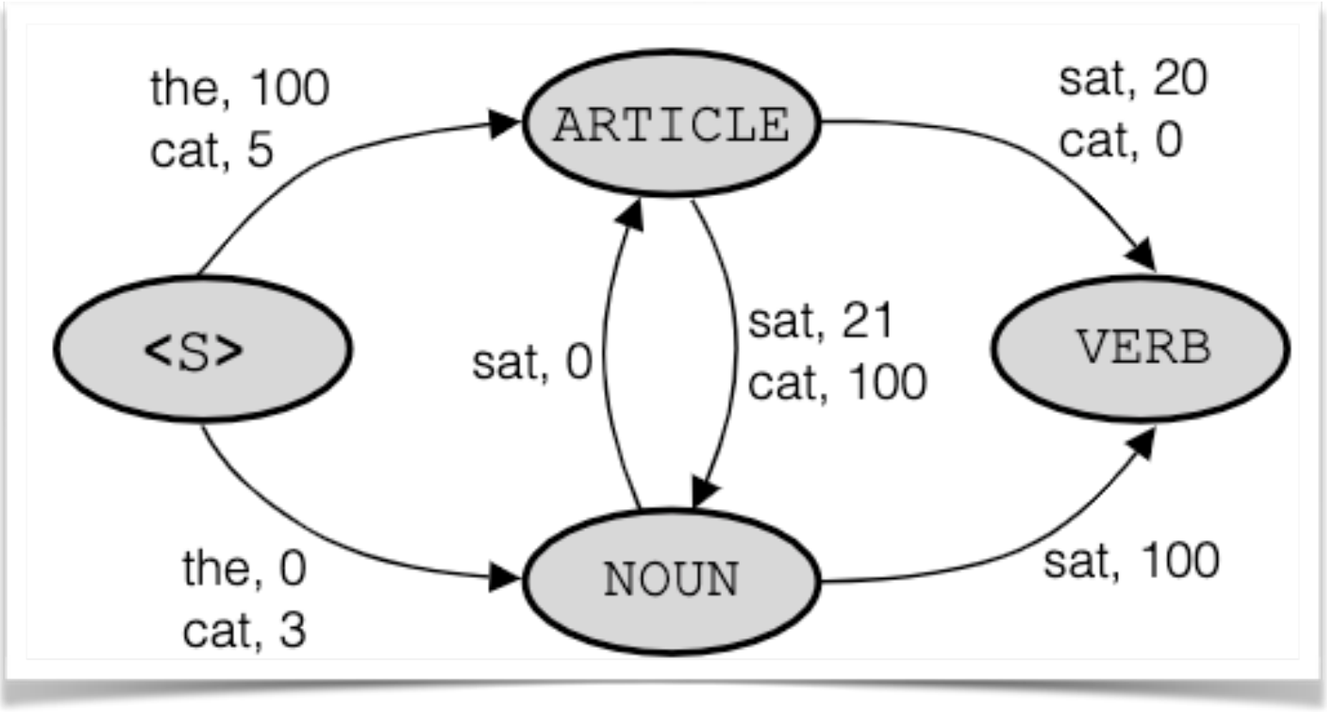
x = cat sat

$$Z(x) = \exp(5+21) + \exp(5+20) + \exp(3+100) + \exp(3,0)$$

A, N A, V N, V N, A

Example from Awni Hannun, https://awni.github.io/label-bias/

For simplicity assume no self-loops



$$p(y|x) = \frac{\exp\{\sum_i s(y_i, x_i, y_{i-1})\}}{\sum_{y'} \exp\{\sum_i s(y'_i, x_i, y'_{i-1})\}}$$

$\mathcal{Y}$ `[ARTICLE, NOUN, VERB]` V [cat, sat, the]

x = cat sat

$$Z(x) = \exp(5+21) + \exp(5+20) + \exp(3+100) + \exp(3,0)$$

A, N A, V N, V N, A

$$p(N, V) \sim 1$$

Example from Awni Hannun, https://awni.github.io/label-bias/

Regularization

$$Z(x) = \exp(5 + 21) + \exp(5 + 20) + \exp(3 + 100) + \exp(3, 0)$$

A, N A, V N, V N, A

$$p(N, V) \sim 1$$

- This suggests maybe not great calibration — scores too large? exp(103) is really big!

- Important to **regularize** parameters

Prediction

$$Z(x) = \underbrace{\exp(5+21)}_{\text{A, N}} + \underbrace{\exp(5+20)}_{\text{A, V}} + \underbrace{\exp(3+100)}_{\text{N, V}} + \underbrace{\exp(3,0)}_{\text{N, A}}$$

$$p(N, V) \sim 1$$

- Do we actually need to compute Z if we just want to make a prediction?

Prediction

$$Z(x) = \underbrace{\exp(5+21)}_{\text{A, N}} + \underbrace{\exp(5+20)}_{\text{A, V}} + \underbrace{\exp(3+100)}_{\text{N, V}} + \underbrace{\exp(3,0)}_{\text{N, A}}$$

$$p(N, V) \sim 1$$

- Do we actually need to compute *Z* if we just want to make a prediction?

- No; we just need *argmax* over *y'*. How can we compute efficiently?

Prediction

$$Z(x) = \underbrace{\exp(5+21)}_{\text{A, N}} + \underbrace{\exp(5+20)}_{\text{A, V}} + \underbrace{\exp(3+100)}_{\text{N, V}} + \underbrace{\exp(3,0)}_{\text{N, A}}$$

$$p(N, V) \sim 1$$

- Do we actually need to compute *Z* if we just want to make a prediction?

- No; we just need *argmax* over *y'.* How can we compute efficiently? **Dynamic programming** (from last time)

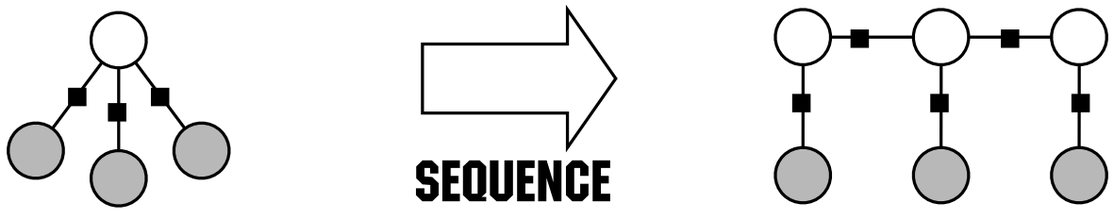
Parameter estimation for *Linear-Chain CRFs* (board)

Example: OCR

https://pystruct.github.io/auto_examples/plot_letters.html

(Notebook)

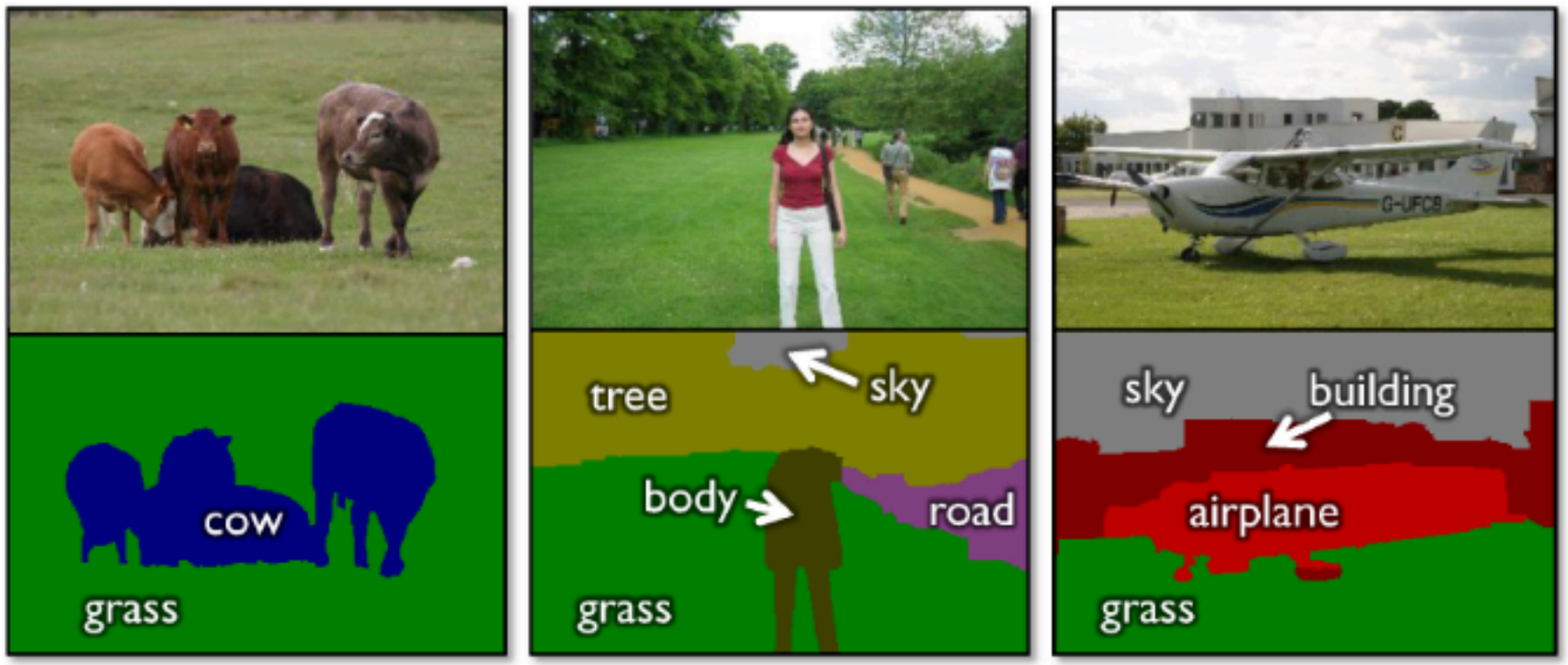
Beyond linear-chains



Logistic Regression

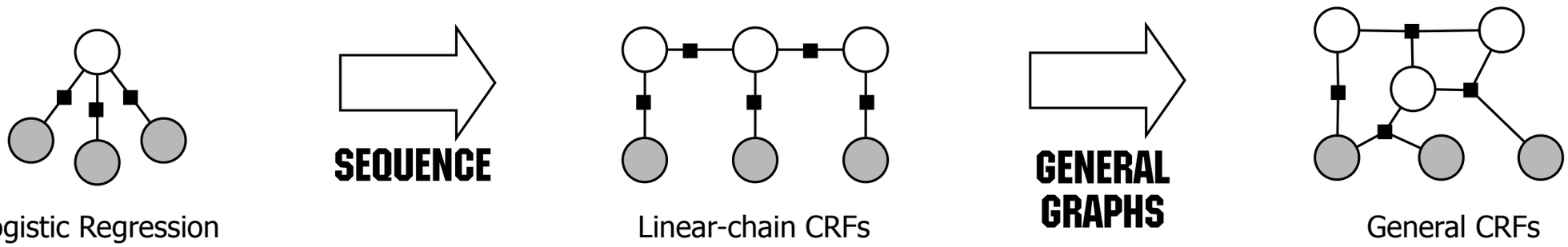
Linear-chain CRFs

Figure from Sutton and McCallum, 2011



Source: Picture taken from "TextonBoost" for Image Understanding: Multiclass Object recognition and Segmentation by jointly modelling Texture, Layout and Context". Jamie Shotton et. al. IJCV 2009

Beyond linear-chains



Logistic Regression | Linear-chain CRFs | General CRFs

Figure from Sutton and McCallum, 2011

Beyond linear-chains

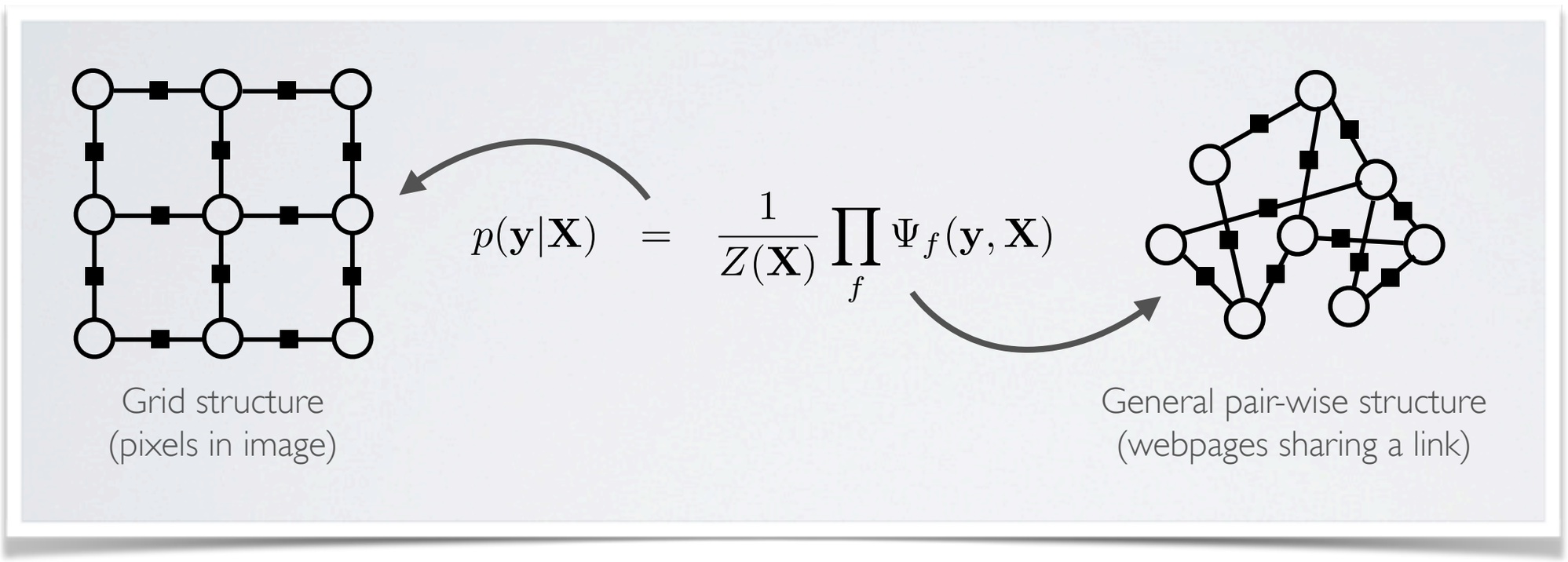


Figure from Hugo Larochelle, http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Training general CRFs

* Here we denote parameters θ instead of **w**

make $y^{(t)}$ more likely

$$\frac{\partial -\log p(\mathbf{y}^{(t)}|\mathbf{X}^{(t)})}{\partial \theta *} = -\overbrace{\left(\sum_f \frac{\partial}{\partial \theta} \log \Psi_f(\mathbf{y}^{(t)}, \mathbf{X}^{(t)})\right.}$$

$$\underbrace{\left. - \mathrm{E}_{\mathbf{y}}\left[\sum_f \frac{\partial}{\partial \theta} \log \Psi_f(\mathbf{y}, \mathbf{X}^{(t)}) \,\middle|\, \mathbf{X}^{(t)}\right]\right)}$$

make everything less likely

Looks similar to what we had for linear-chain, but can no longer use dynamic programming to efficiently take expectation over **y**

Figure from Hugo Larochelle, http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Summary: Structured prediction

- When labels *y* are *correlated* (and where for a given instance *x* and *y* are both tensors) structured prediction models attempt to exploit this

Summary: Structured prediction

- When labels *y* are *correlated* (and where for a given instance *x* and *y* are both tensors) structured prediction models attempt to exploit this

- Hidden Markov Models (HMMs) are a *generative* approach that model P(x, y)

Summary: Structured prediction

- When labels *y* are *correlated* (and where for a given instance *x* and *y* are both tensors) structured prediction models attempt to exploit this

- Hidden Markov Models (HMMs) are a *generative* approach that model P(x, y)

- Structured perceptrons, MEMMs, and CRFs are *conditional* models model p(y|x)

Summary: Structured prediction

- When labels *y* are *correlated* (and where for a given instance *x* and *y* are both tensors) structured prediction models attempt to exploit this

- Hidden Markov Models (HMMs) are a *generative* approach that model P(x, y)

- Structured perceptrons, MEMMs, and CRFs are *conditional* models model p(y|x)

- For all we use dynamic programming (Viterbi) for efficient argmaxing, and variants of this to efficiently compute normalization constants, etc.