ANTONIS ANASTASOPOULOS
CS499 INTRODUCTION TO NLP

CONDITIONAL RANDOM FIELDS

https://cs.gmu.edu/~antonis/course/cs499-spring21/

With adapted slides by Graham Neubig

STRUCTURE OF THIS LECTURE

1. Structured Prediction
2. Structured Perceptron
3. Conditional Random Fields
4. Viterbi

A PREDICTION PROBLEM

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

TYPES OF PREDICTION

Two classes (**binary classification**)

I hate this movie → positive / negative

- Multiple classes (**multi-class classification**)

I hate this movie → very good / good / neutral / bad / very bad

- Exponential/infinite labels (**structured prediction**)

I hate this movie → PRP VBP DT NN

I hate this movie → *kono eiga ga kirai*

WHY CALL IT "STRUCTURED" PREDICTION?

Classes are too numerous to enumerate

Need some sort of method to exploit the *problem structure* to learn efficiently

Example of "structure", the following two outputs are similar:

PRP VBP DT NN

PRP VBP VBP NN

MANY VARIETIES OF STRUCTURED PREDICTION!

Models:

RNN-based decoders

Convolution/self attentional decoders

CRFs w/ local factors

Training algorithms:

Structured perceptron, structured large margin

Sampling corruptions of data

Exact enumeration with dynamic programs

Reinforcement learning/minimum risk training

SEQUENCE LABELING

One tag for one word

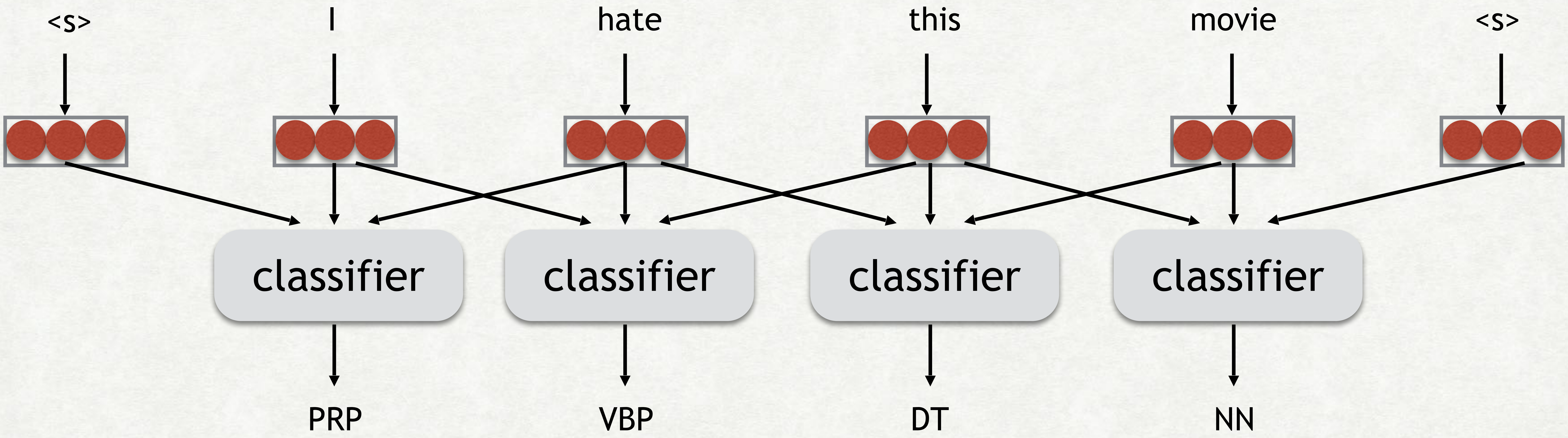
e.g. Part of speech tagging

I	hate	this	movie
↓	↓	↓	↓
PRP	VBP	DT	NN

- e.g. Named entity recognition

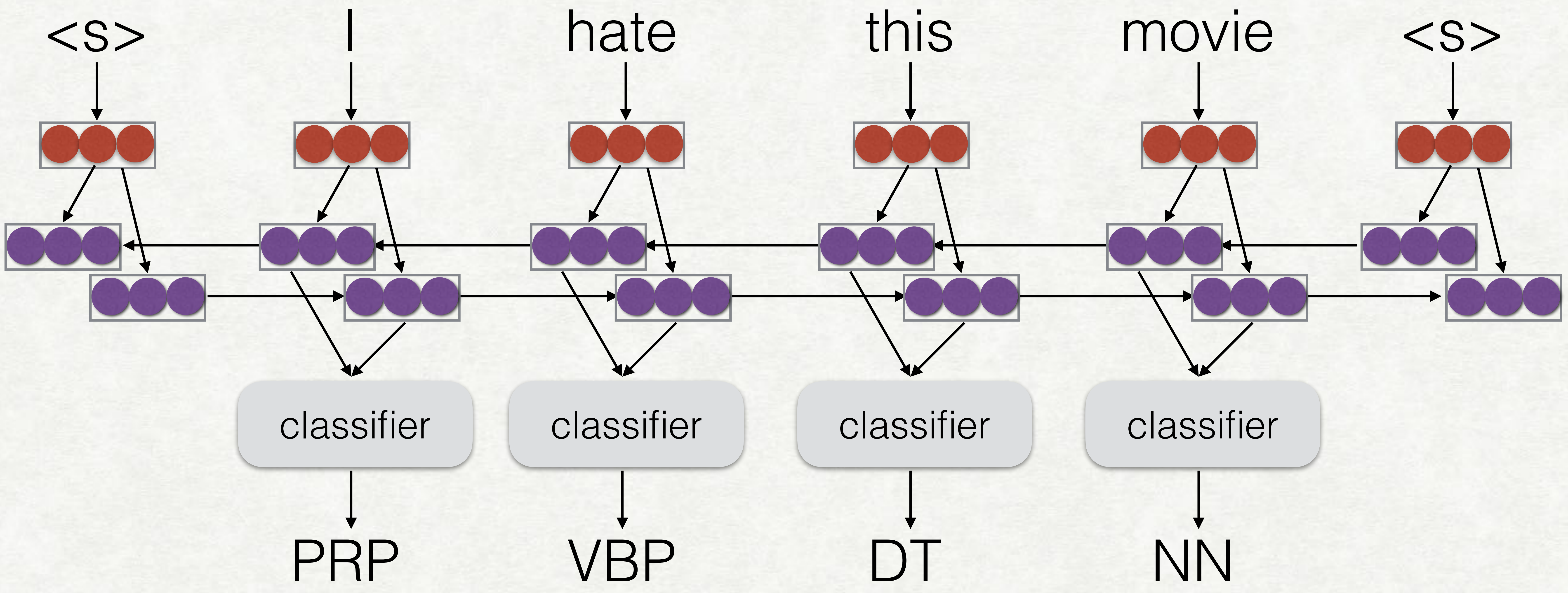
The	movie	featured	Keanu	Reeves
↓	↓	↓	↓	↓
O	O	O	B-PER	I-PER

SEQUENCE LABELING AS INDEPENDENT CLASSIFICATION



SEQUENCE LABELING W/ BILSTM

Still not modeling output structure! Outputs are independent



WHY MODEL INTERACTIONS IN OUTPUT?

Consistency is important!

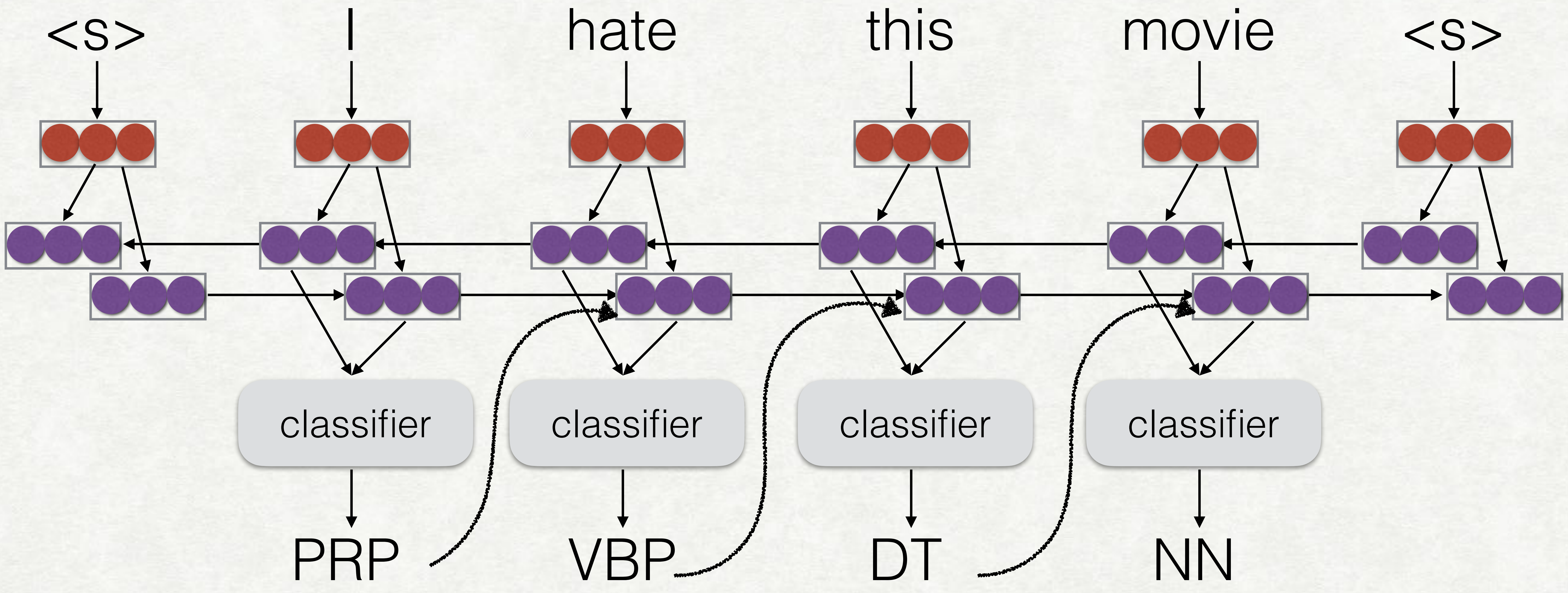
time	flies	like	an	arrow	
N	V	Prep	DT	N	(time moves similarly to an arrow)
N	NS	V	DT	N	("time flies" are fond of arrows)
V	N	Prep	DT	N	(please measure the time of flies similarly to how an arrow would)

↓ max frequency

N	NS	Prep	DT	N	("time flies" that are similar to an arrow)

A TAGGER CONSIDERING OUTPUT STRUCTURE

Tags are inter-dependent



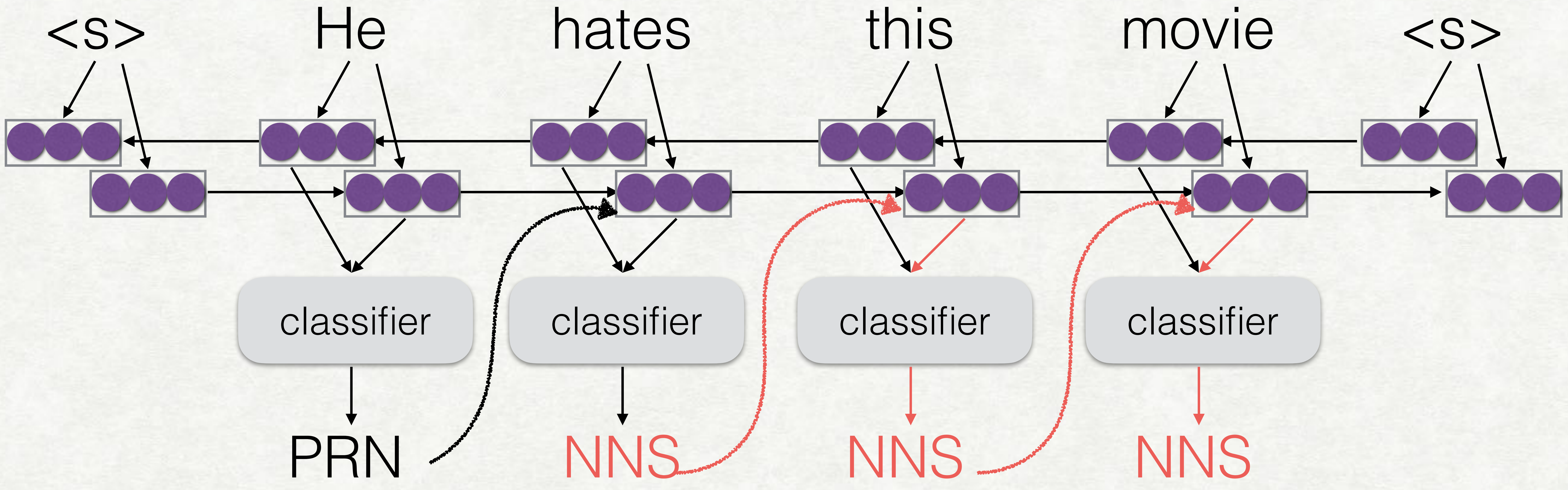
TRAINING STRUCTURED MODELS

Simplest training method "teacher forcing"

Just feed in the *correct* previous tag

TEACHER FORCING AND EXPOSURE BIAS

Teacher forcing assumes feeding correct previous input, but at test time we may make mistakes that propagate



- **Exposure bias:** The model is not exposed to mistakes during training, and cannot deal with them at test

LOCAL NORMALIZATION VS. GLOBAL NORMALIZATION

Locally normalized models: each decision made by the model has a probability that adds to one

$$P(Y \mid X) = \prod_{j=1}^{|Y|} \frac{e^{S(y_j|X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{y}_j \in V} e^{S(\tilde{y}_j|X,y_1,\ldots,y_{j-1})}}$$

Globally normalized models (a.k.a. energy-based models): each sentence has a score, which is not normalized over a particular decision

$$P(Y \mid X) = \frac{e^{\sum_{j=1}^{|Y|} S(y_j|X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{Y} \in V*} e^{\sum_{j=1}^{|\tilde{Y}|} S(\tilde{y}_j|X,\tilde{y}_1,\ldots,\tilde{y}_{j-1})}}$$

PROBLEMS TRAINING GLOBALLY NORMALIZED MODELS

Problem: the denominator is too big to expand naively

We must do something tricky:

$$P(Y \mid X) = \frac{e^{\sum_{j=1}^{|Y|} S(y_j|X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{Y} \in V*} e^{\sum_{j=1}^{|\tilde{Y}|} S(\tilde{y}_j|X,\tilde{y}_1,\ldots,\tilde{y}_{j-1})}}$$

Consider only a subset of hypotheses

Design the model so we can efficiently enumerate all hypotheses

STRUCTURED PERCEPTRON

THE STRUCTURED PERCEPTRON ALGORITHM

An extremely simple way of training (non-probabilistic) global models

Find the one-best, and if it's score is better than the correct answer, adjust parameters to fix this

$$\hat{Y} = \text{argmax}_{\tilde{Y} \neq Y} S(\tilde{Y} \mid X; \theta)$$

Find one best

$$\textbf{if } S(\hat{Y} \mid X; \theta) \geq S(Y \mid X; \theta) \textbf{ then}$$

If score better than reference

$$\theta \leftarrow \theta + \alpha\left(\frac{\partial S(Y|X;\theta)}{\partial \theta} - \frac{\partial S(\hat{Y}|X;\theta)}{\partial \theta}\right)$$

Increase score of ref, decrease score of one-best (here, SGD update)

$$\textbf{end if}$$

STRUCTURED PERCEPTRON LOSS

Structured perceptron can also be expressed as a loss function!

$$\ell_{\text{percept}}(X, Y) = \max(0, S(\hat{Y} \mid X; \theta) - S(Y \mid X; \theta))$$

- Resulting **gradient looks like perceptron algorithm**

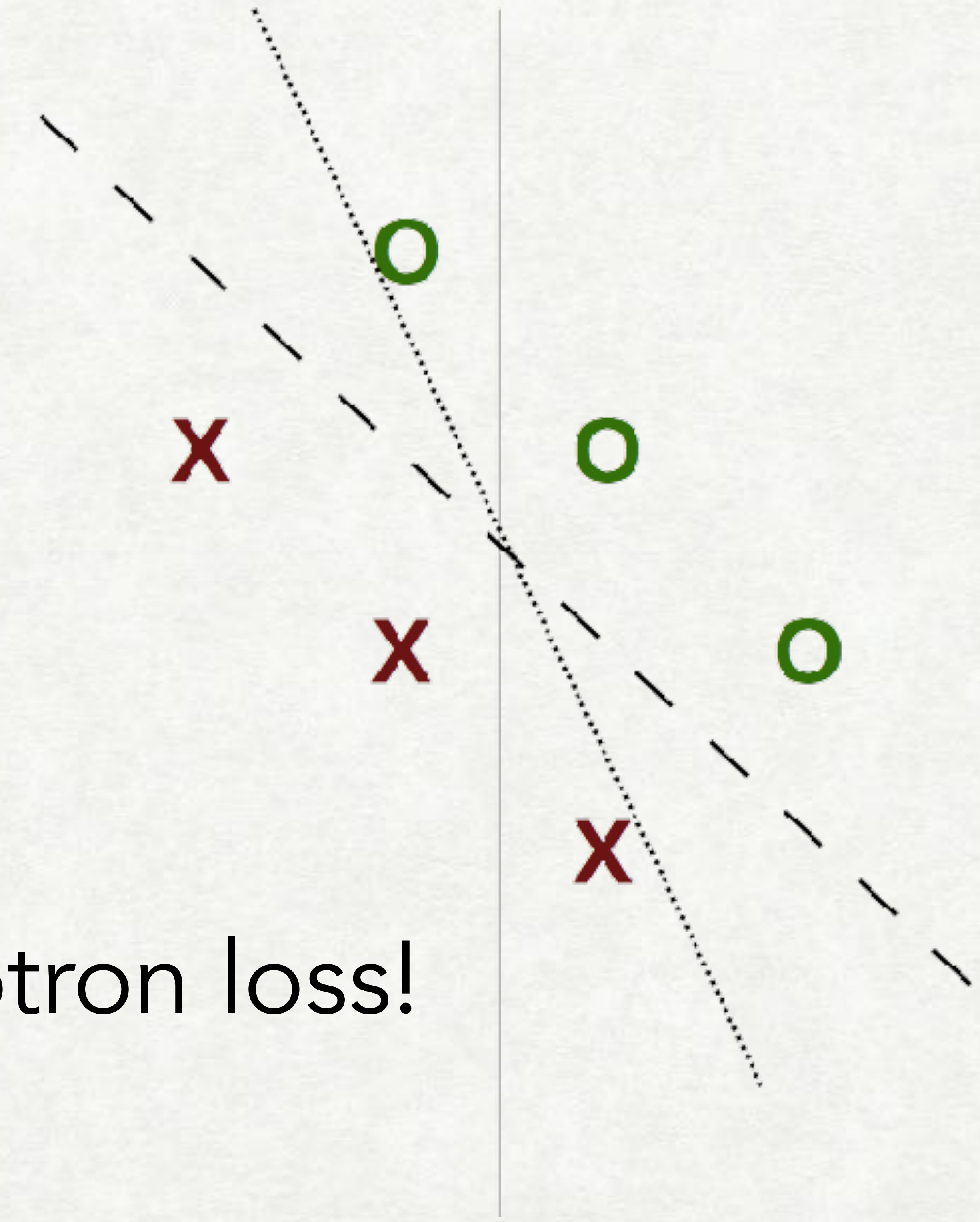
$$\frac{\partial \ell_{\text{percept}}(X, Y; \theta)}{\partial \theta} = \begin{cases} \frac{\partial S(Y|X;\theta)}{\partial \theta} - \frac{\partial S(\hat{Y}|X;\theta)}{\partial \theta} & \text{if } S(\hat{Y} \mid X; \theta) \geq S(Y \mid X; \theta) \\ 0 & \text{otherwise} \end{cases}$$

- This is a normal loss function, **can be used in NNs**
- But! Requires finding the argmax in addition to the true candidate: must **do prediction during training**

HINGE LOSS AND COST-SENSITIVE TRAINING

PERCEPTRON AND UNCERTAINTY

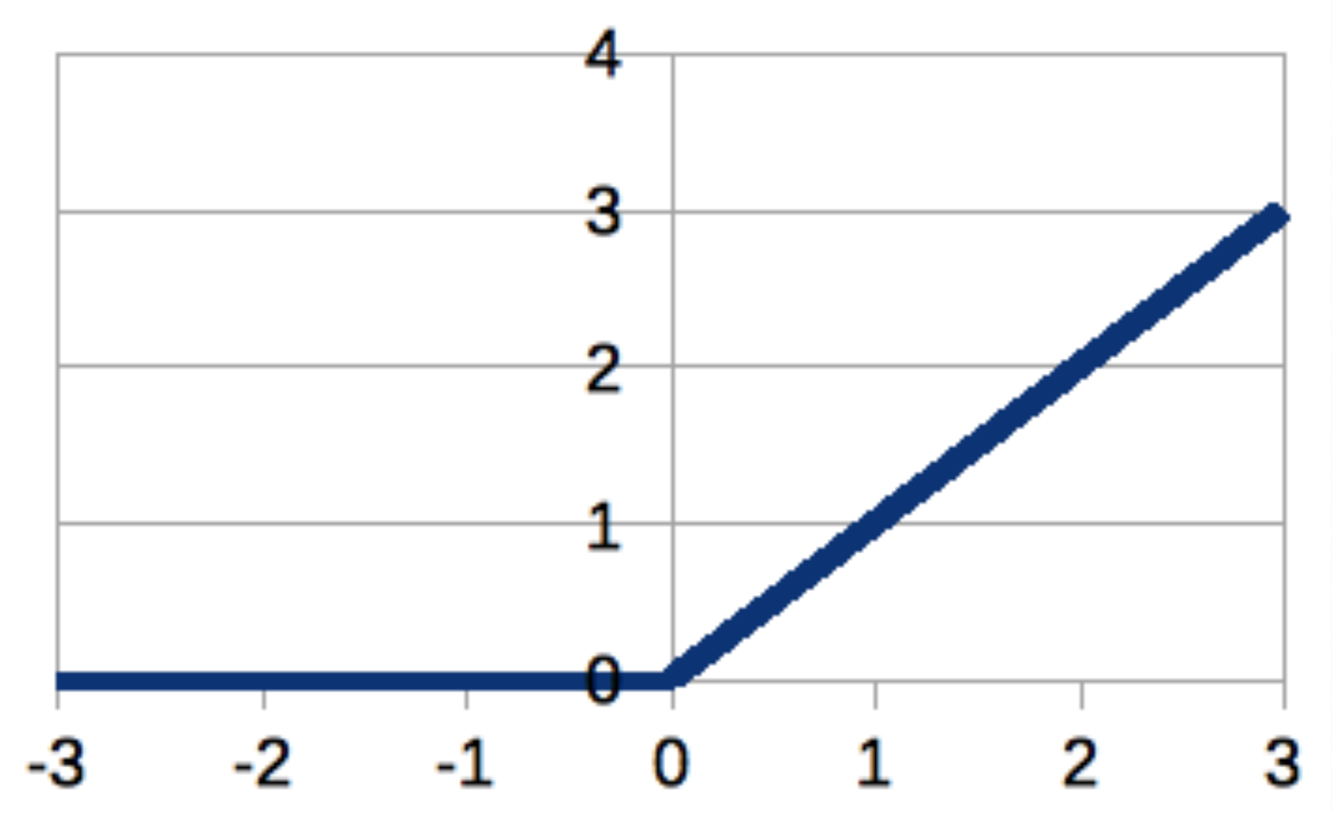
Which is better, dotted or dashed?



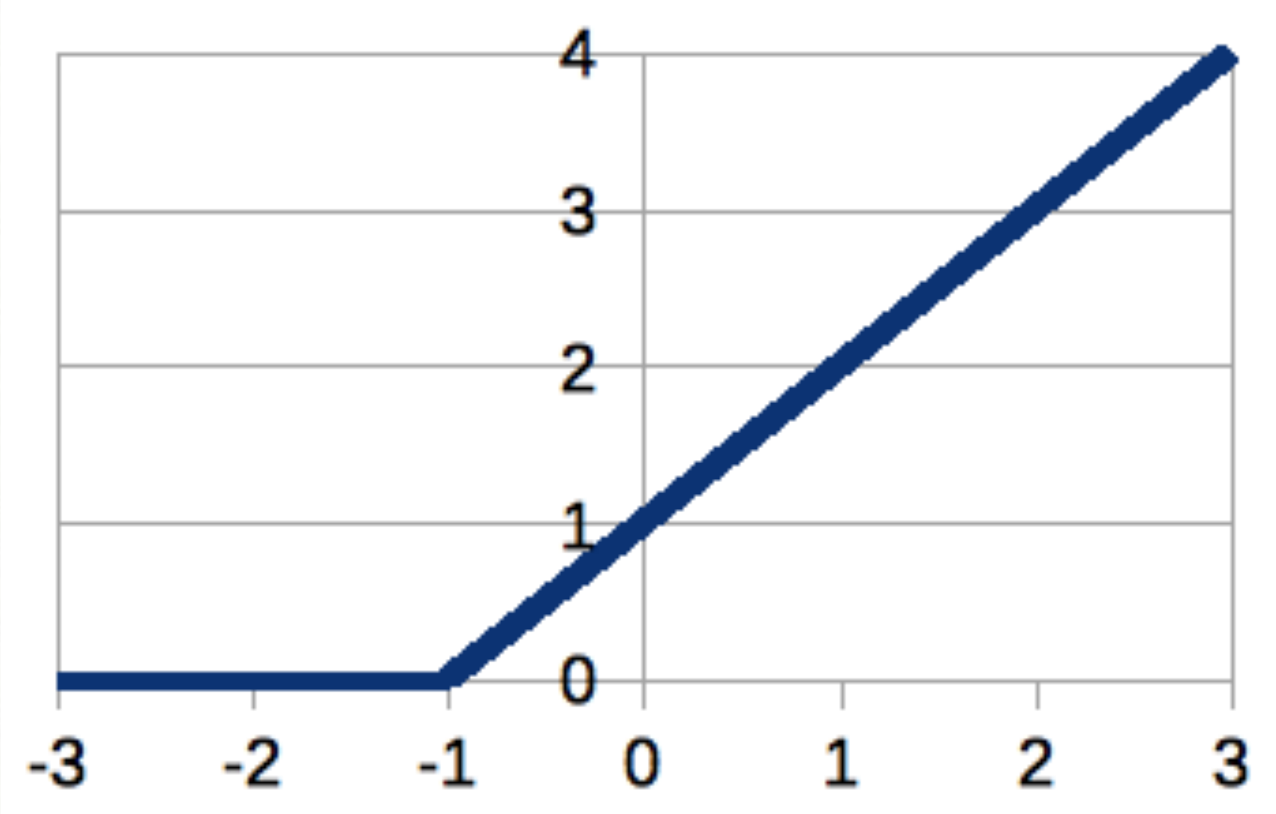
Both have zero perceptron loss!

ADDING A "MARGIN" WITH HINGE LOSS

Penalize when incorrect answer is within margin *m*



Perceptron

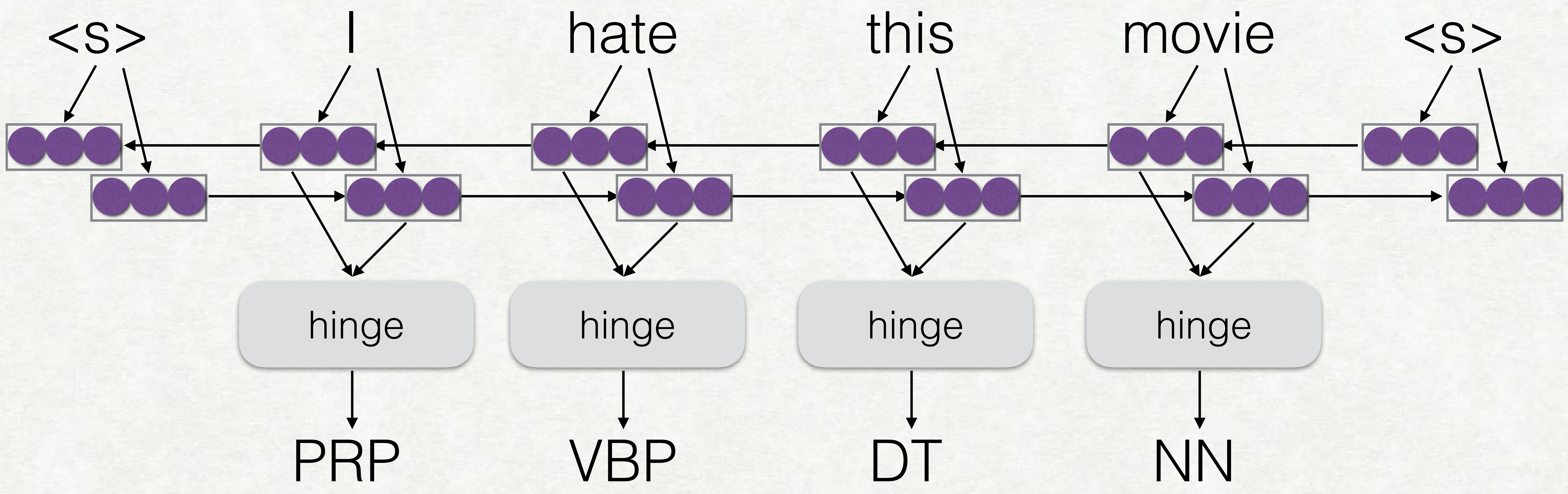


Hinge

$$\ell_{\text{hinge}}(x, y; \theta) = \max(0, m + S(\hat{y} \mid x; \theta) - S(y \mid x; \theta))$$

HINGE LOSS FOR ANY CLASSIFIER!

We can swap cross-entropy for hinge loss anytime




```
loss = dy.pickneglogsoftmax(score, answer)
                    ↓
loss = dy.hinge(score, answer, m=1)
```

LOCALLY-DEPENDENT MODELS

PROBLEMS

- Independent classification models
  - Strong independent assumption

$$P(Y|X) = \prod_{i=1}^{L} P(y_i|X)$$

  - No guarantee of valid (consistent) structured outputs
    - BIO tagging scheme in NER
- Locally normalized models (e.g. history-based RNN, seq2seq)
  - Prior order

$$P(Y|X) = \prod_{i=1}^{L} P(y_i|X, y_{<i})$$

  - Approximating decoding
    - Greedy search
    - Beam search
  - Label bias

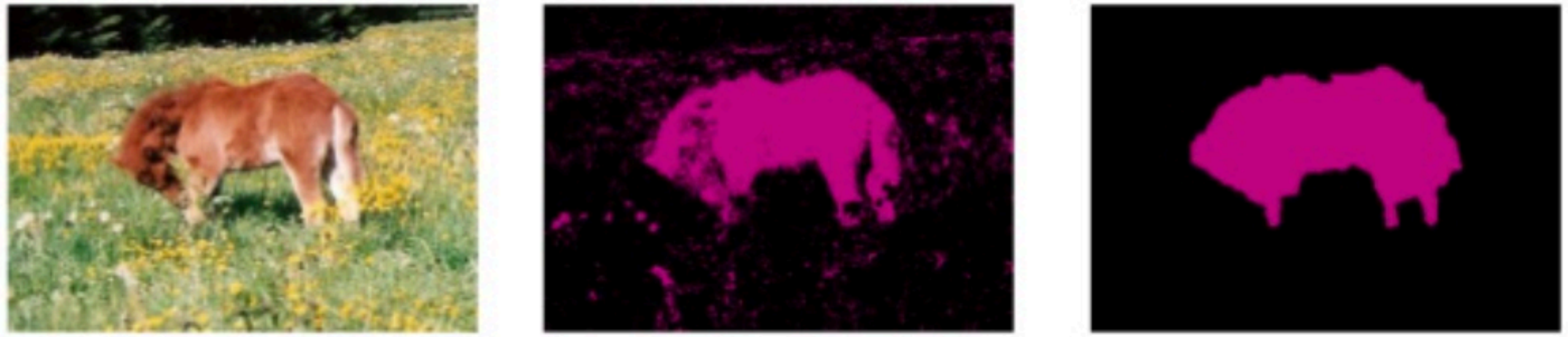
MODELS W/ LOCAL DEPENDENCIES

Some independence assumptions, but not entirely independent (local dependencies)

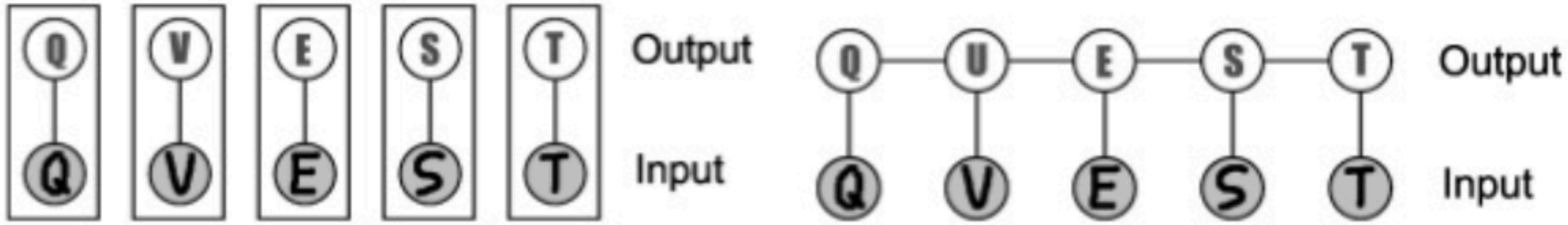
Exact and optimal decoding/training via dynamic programs

Conditional Random Fields!
(CRFs)

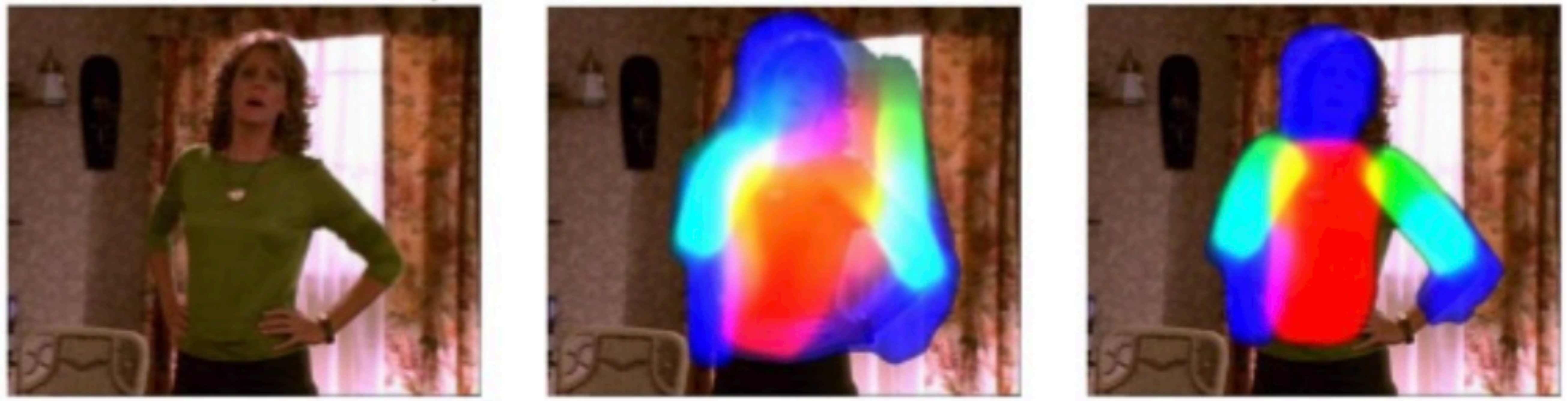
LOCAL VS LOCALLY NORMALIZED



original   local classification   local + smoothness



local classification   local + "correction"



original   local classification   local + geometry

REMINDER: GLOBALLY NORMALIZED MODELS

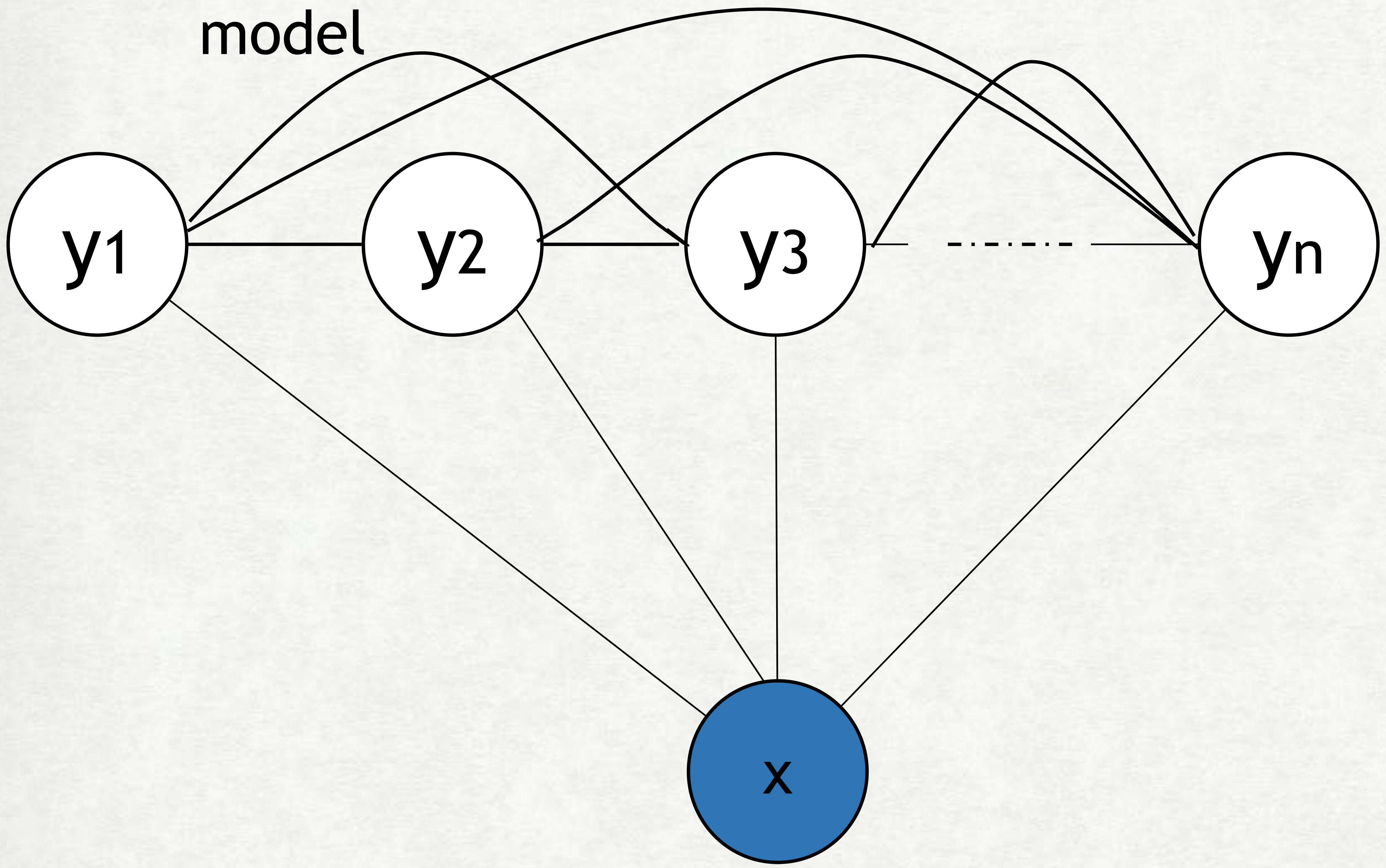
- Each output sequence has a score, which is not normalized over a particular decision

$$P(Y|X) = \frac{\exp(S(Y,X))}{\sum_{Y'} \exp(S(Y',X))} = \frac{\psi(Y,X)}{\sum_{Y'} \psi(Y',X)}$$

where $\psi(Y,X)$ are potential functions.

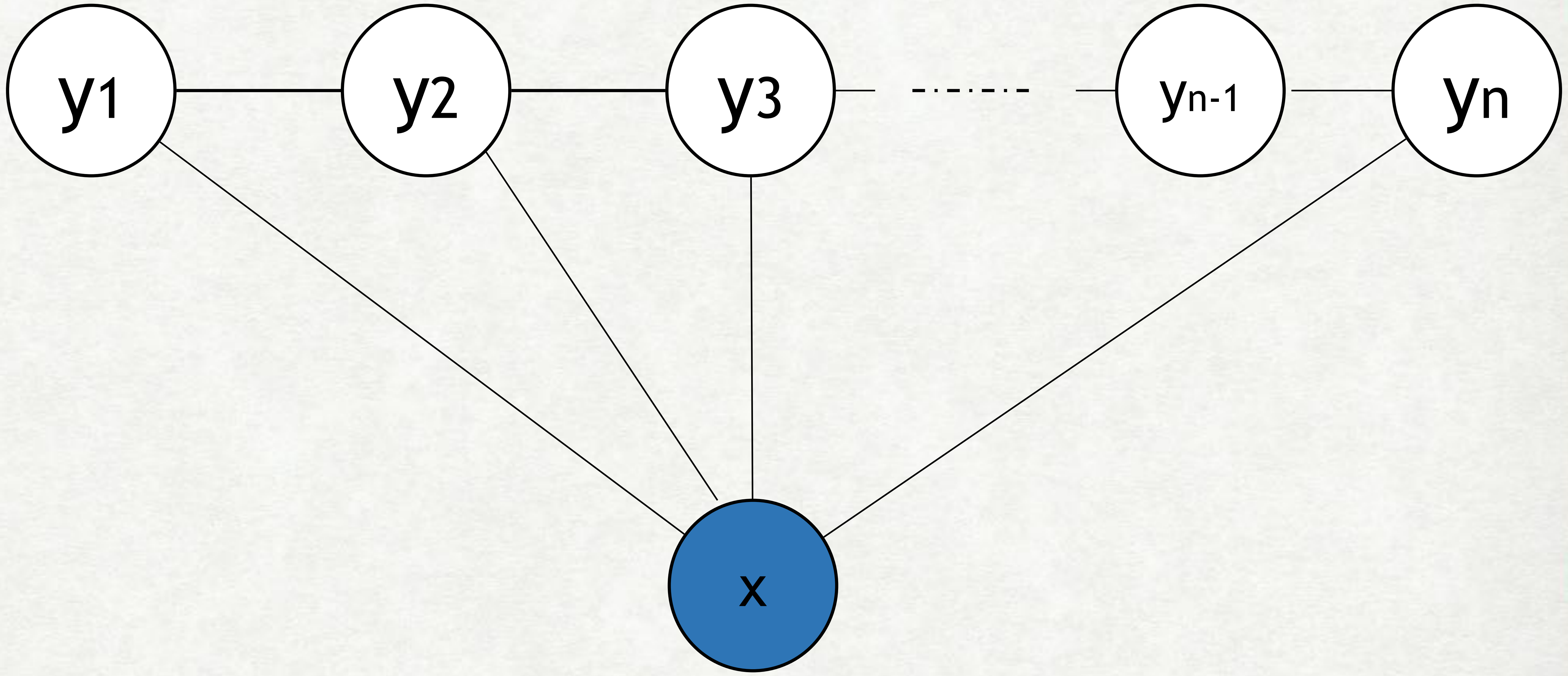
CONDITIONAL RANDOM FIELDS

General form of globally normalized model



$$P(Y|X) = \frac{\psi(Y,X)}{\sum_{Y'} \psi(Y',X)}$$

First-order linear CRF



$$P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)}$$

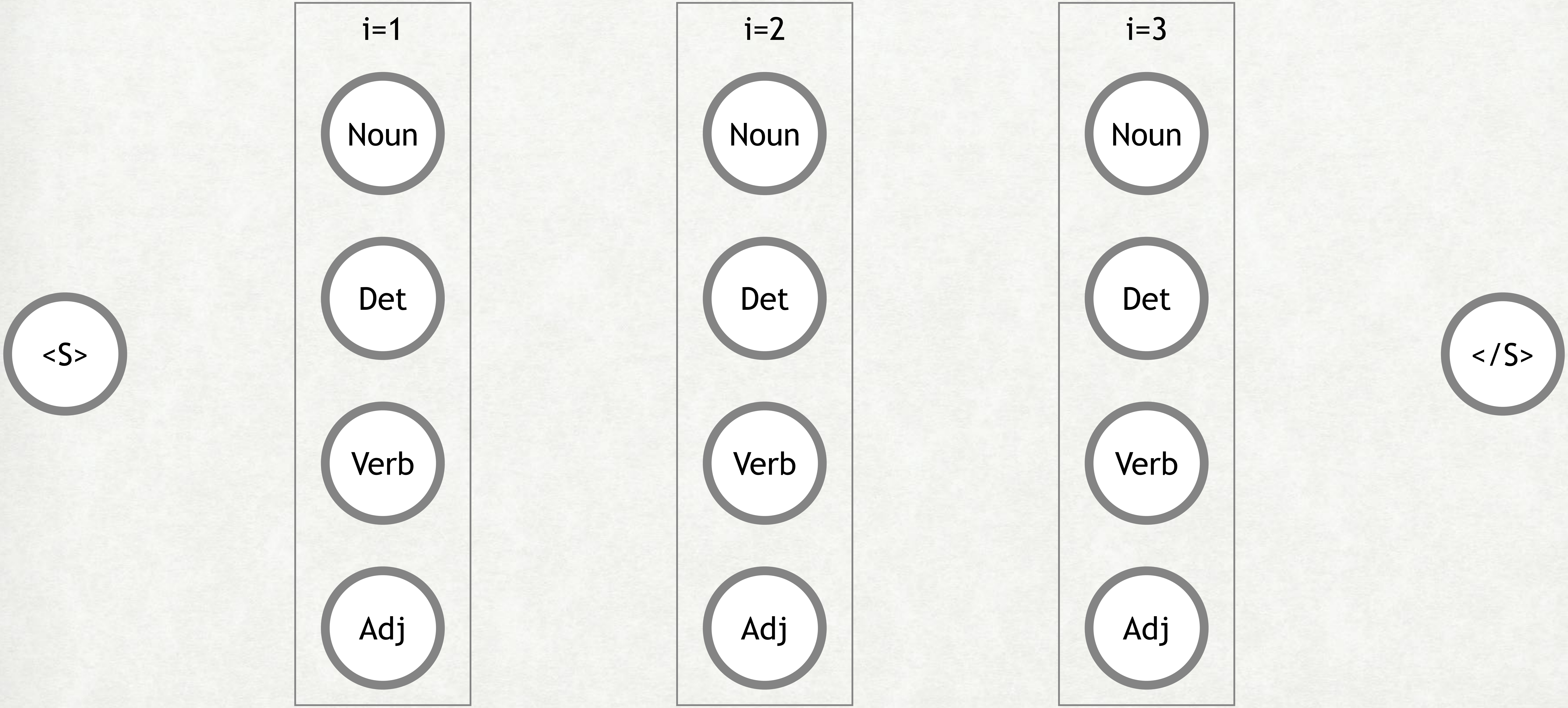
POTENTIAL FUNCTIONS

$$\psi_i(y_{i-1}, y_i, X) = \underbrace{\boxed{t(y_{i-1}, y_i, X)}}_{\text{“Transition”}} \times \underbrace{\boxed{e(y_i, X)}}_{\text{“Emission”}}$$

Simpler Version

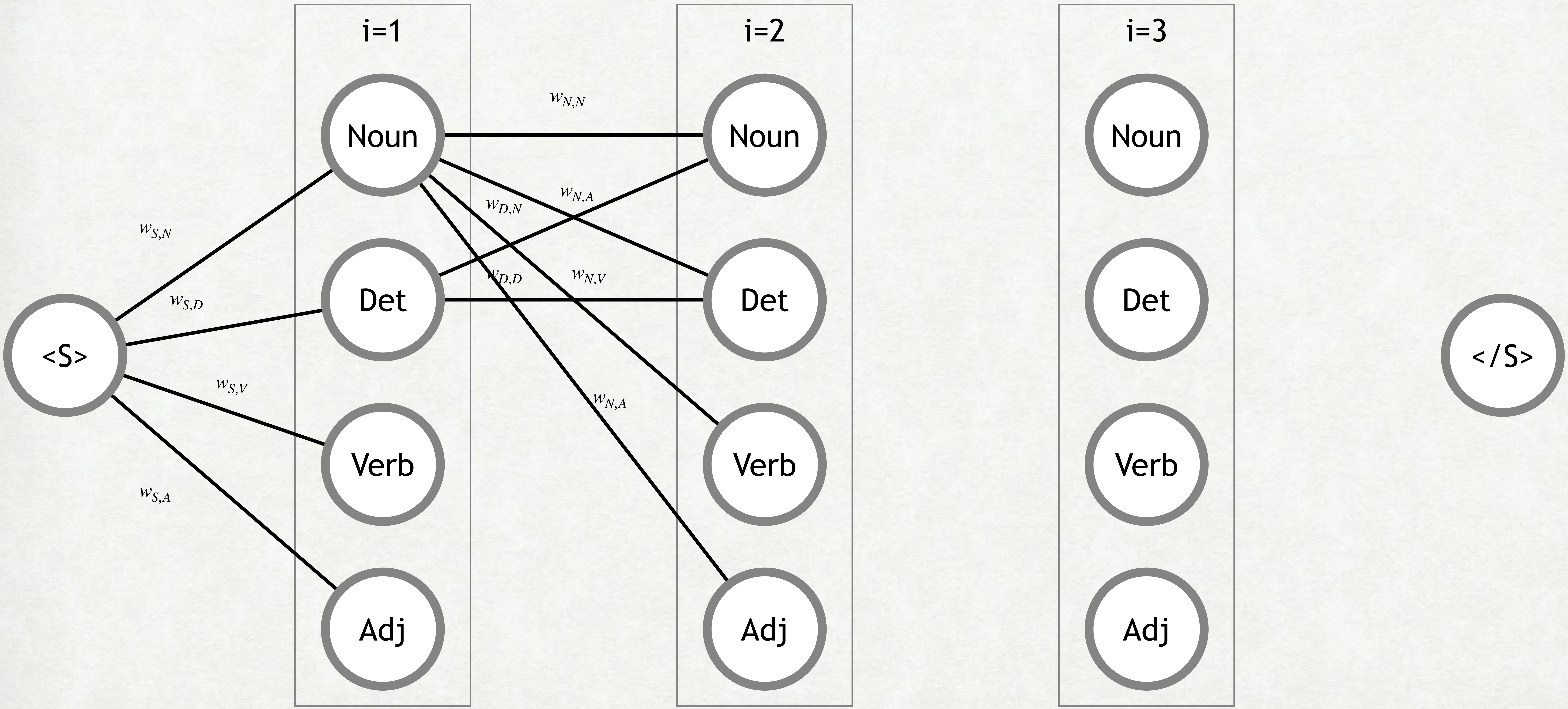
$$\psi_i(y_{i-1}, y_i, X) = t(y_{i-1}, y_i) \times e(y_i, X)$$

EXAMPLE

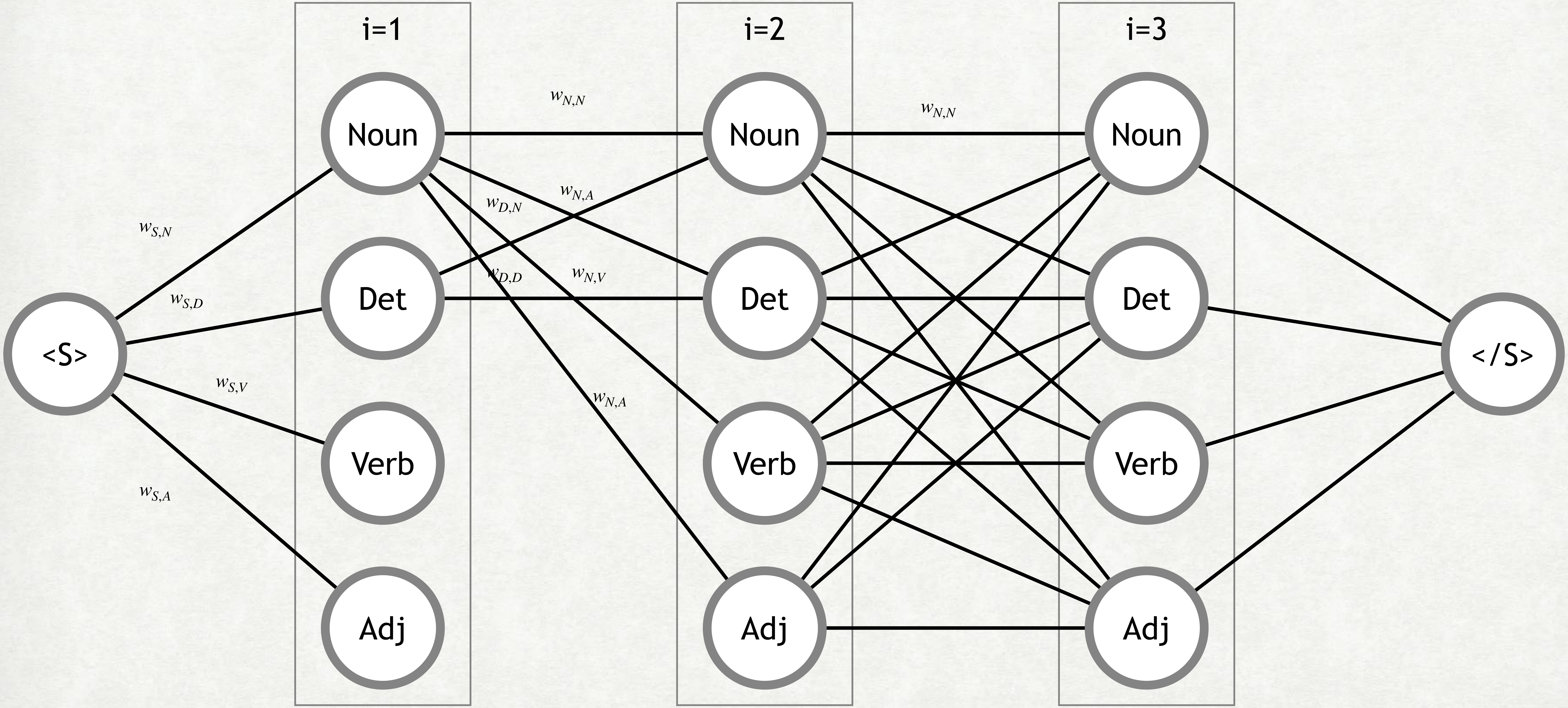


<S>
i=1
Noun
Det
Verb
Adj
i=2
Noun
Det
Verb
Adj
i=3
Noun
Det
Verb
Adj
</S>

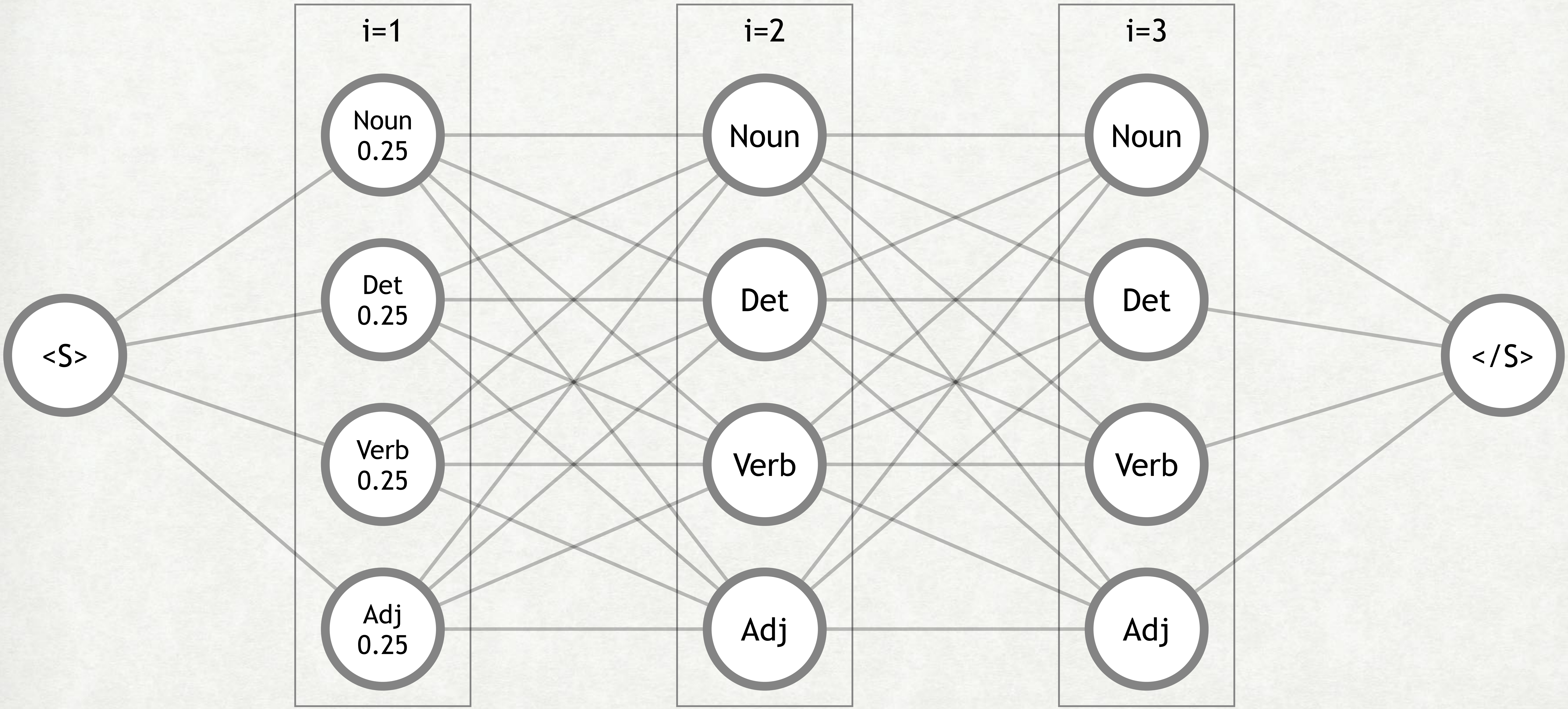
TRANSITION PARAMETERS



TRANSITION PARAMETERS

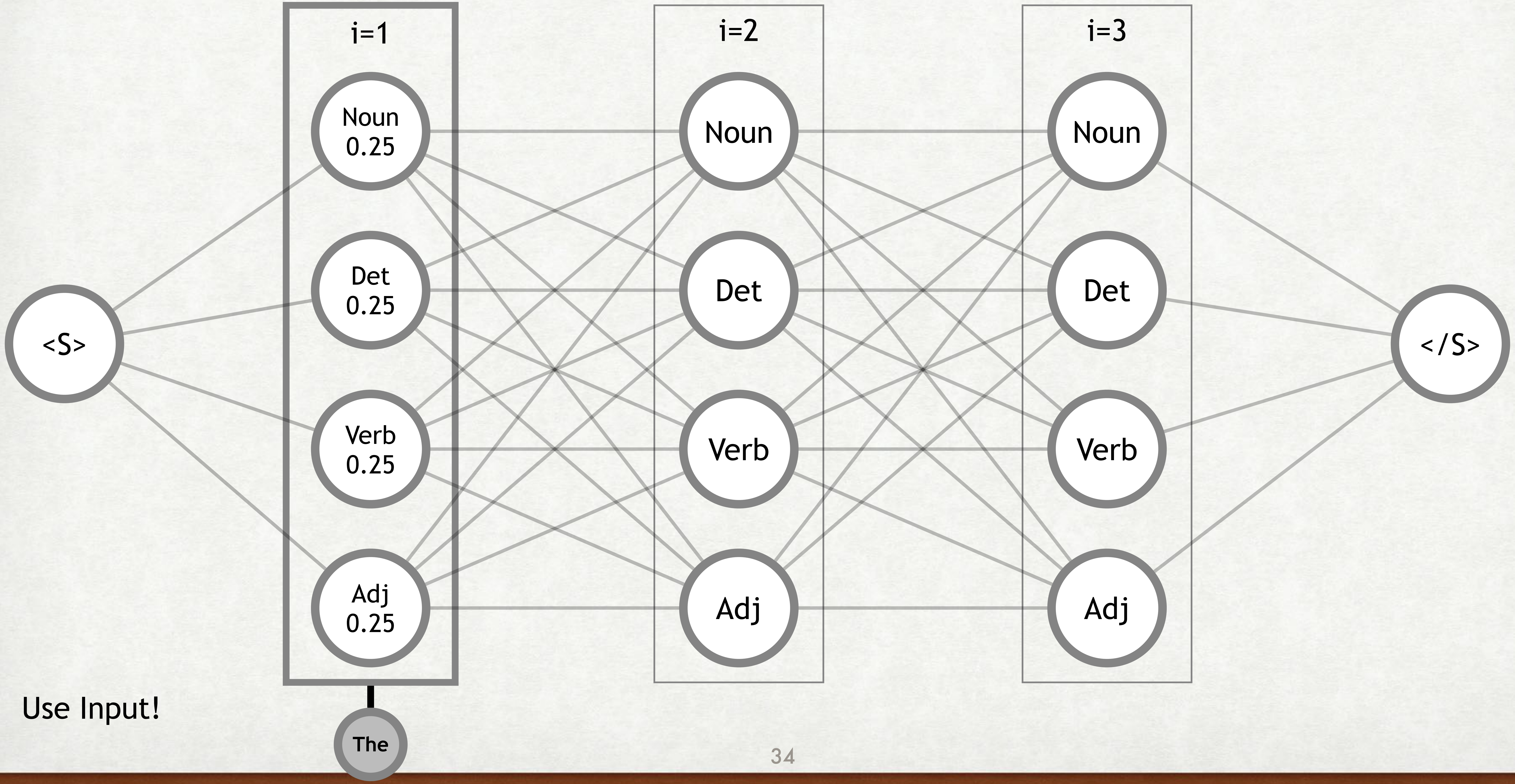


EMISSION PROBABILITIES



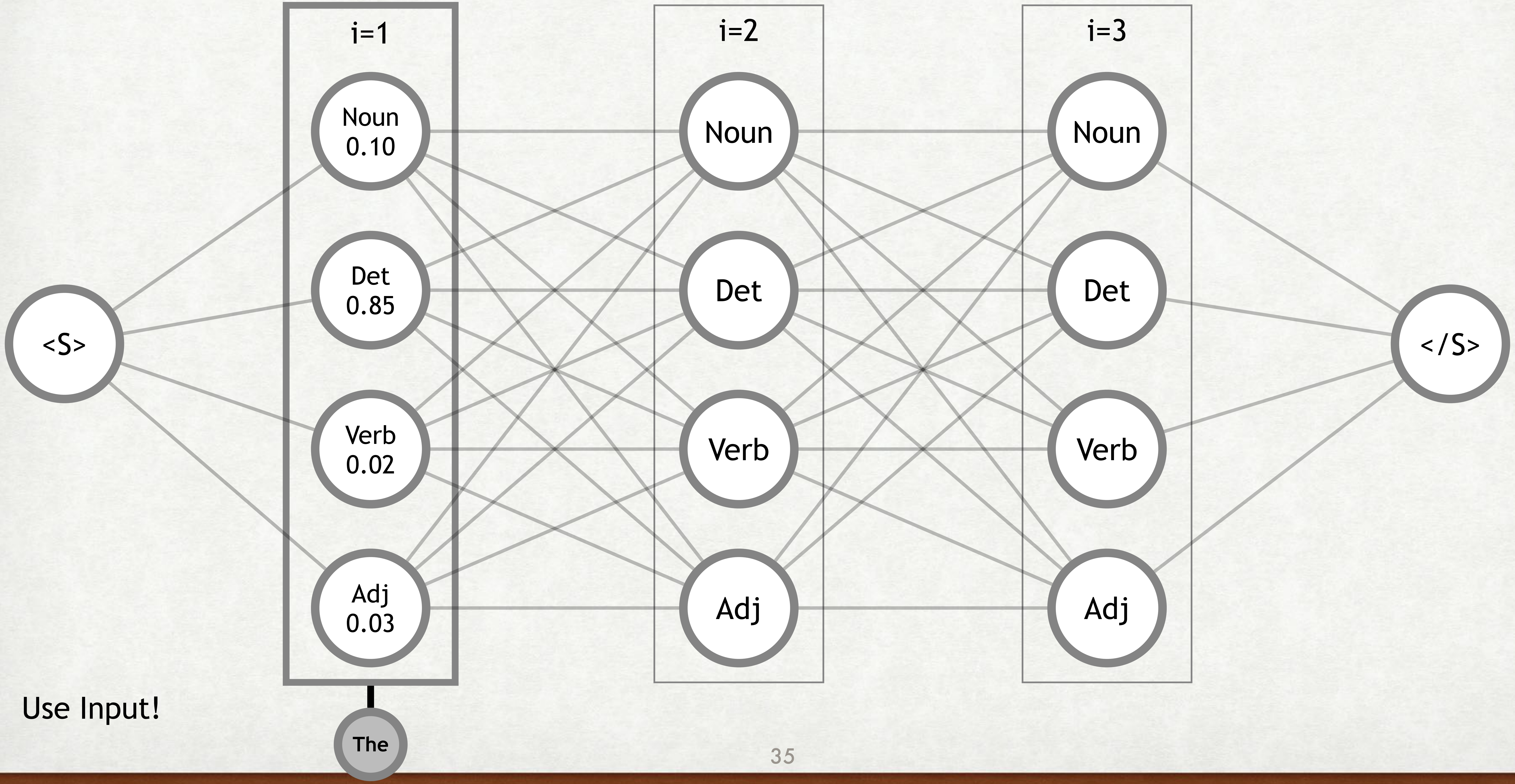
Equally likely?

EMISSION PROBABILITIES



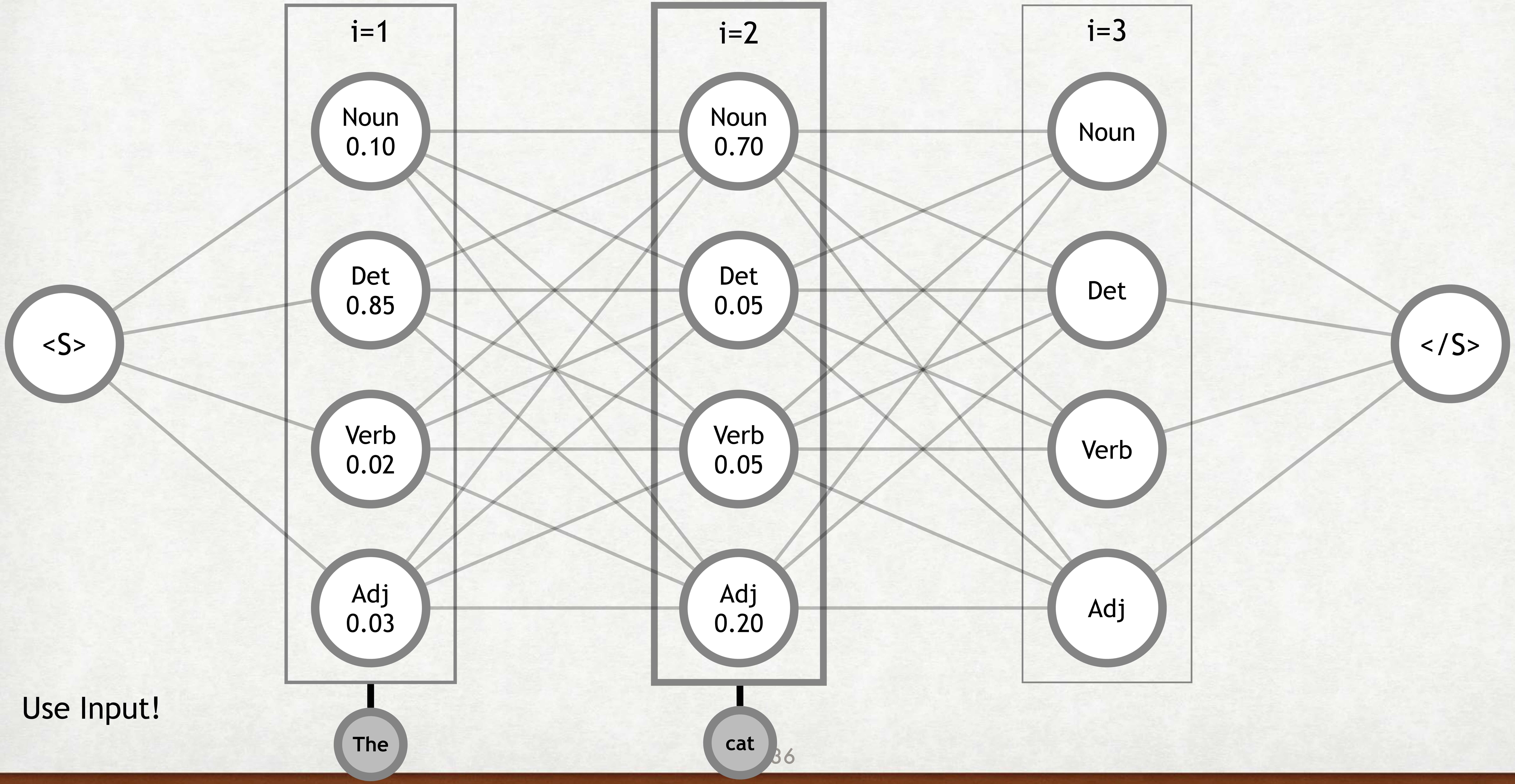
Use Input!

EMISSION PROBABILITIES



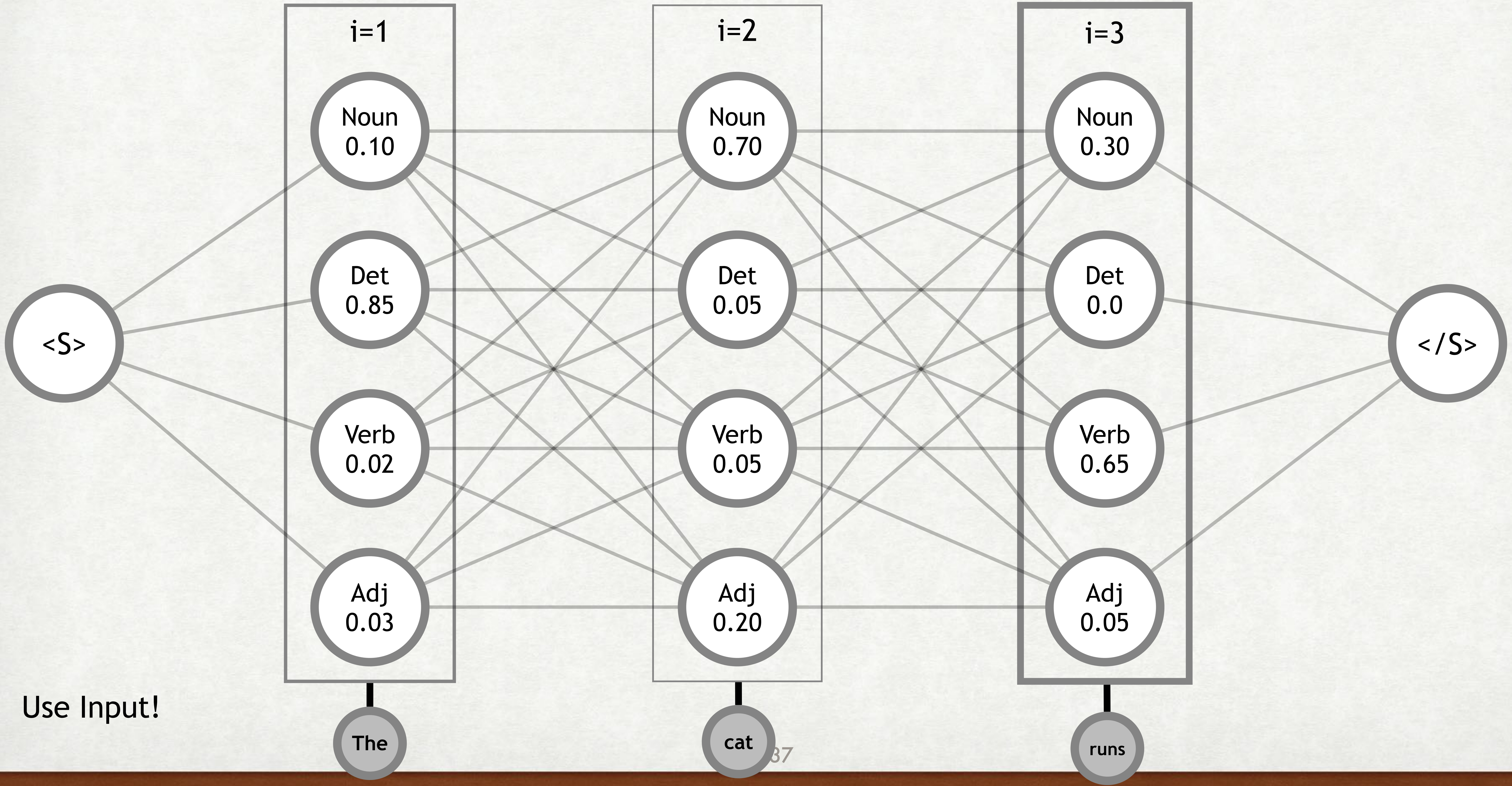
Use Input!

EMISSION PROBABILITIES



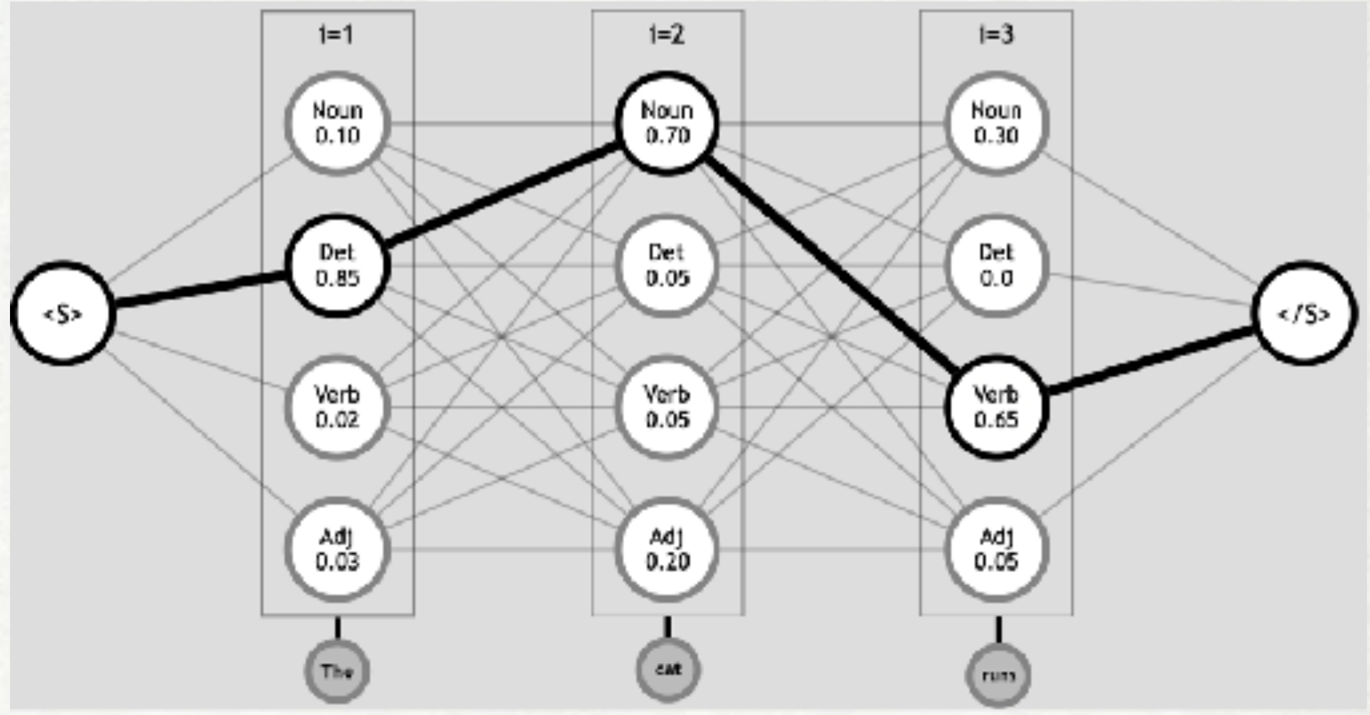
Use Input!

EMISSION PROBABILITIES



Use Input!

TRAINING



We want to maximize the probability
of the correct output sequence

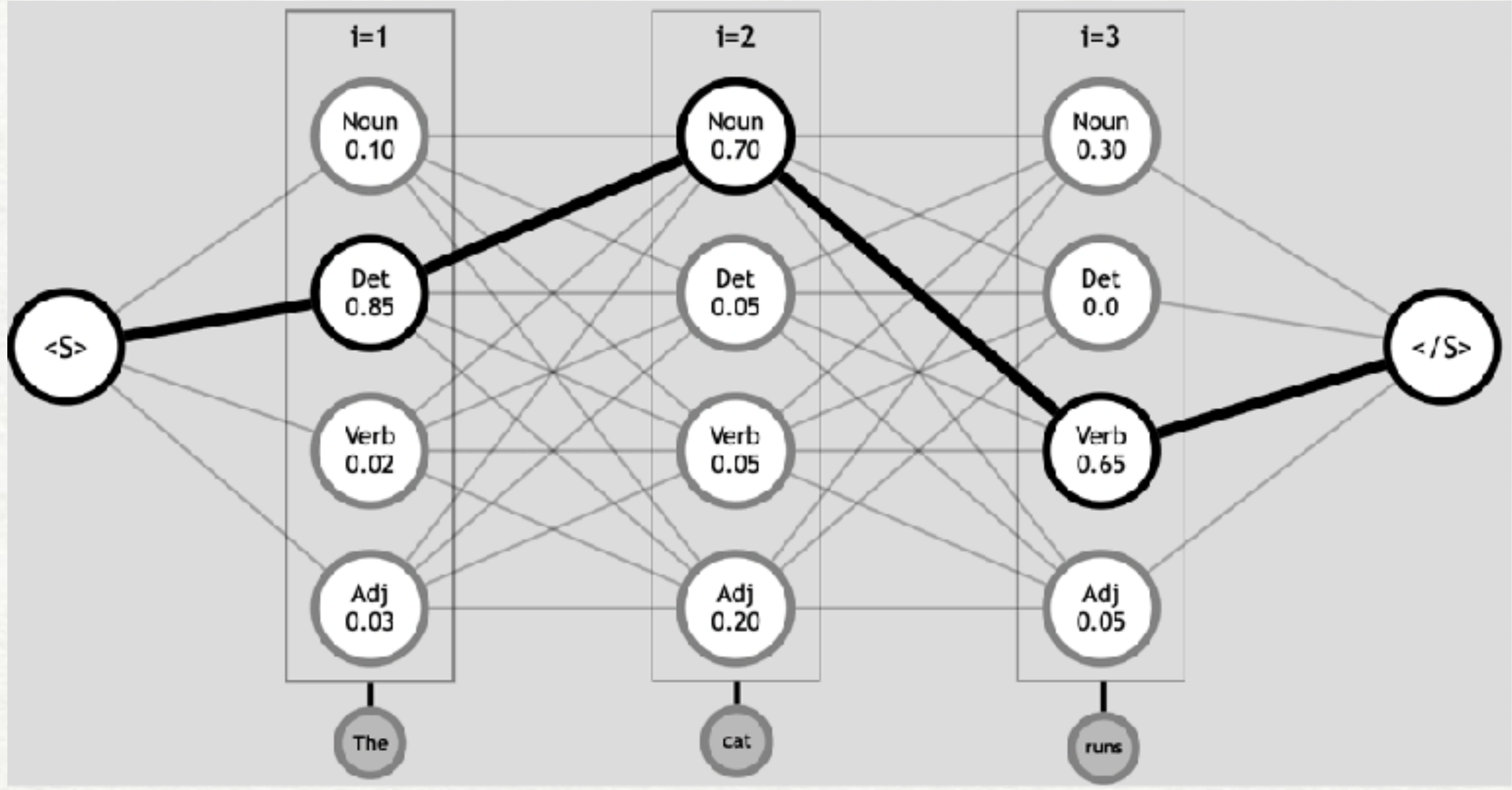
- $P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)} = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{Z(X)}$
- Training: computing the partition function Z(X)

$$Z(X) = \sum_{Y} \prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)$$

- Decoding

$$y^* = argmax_Y P(Y|X)$$

TRAINING



We want to maximize the probability of the correct output sequence

Traditionally:

1. Extract features from the input words/context/sentence
2. Train with features as input, target sequences as desired output
3. Use some optimization technique to weight the feature importance for each position

A bunch of algorithms are nicely implemented in scikit-learn:

https://sklearn-crfsuite.readthedocs.io/en/latest/

TRY IT!

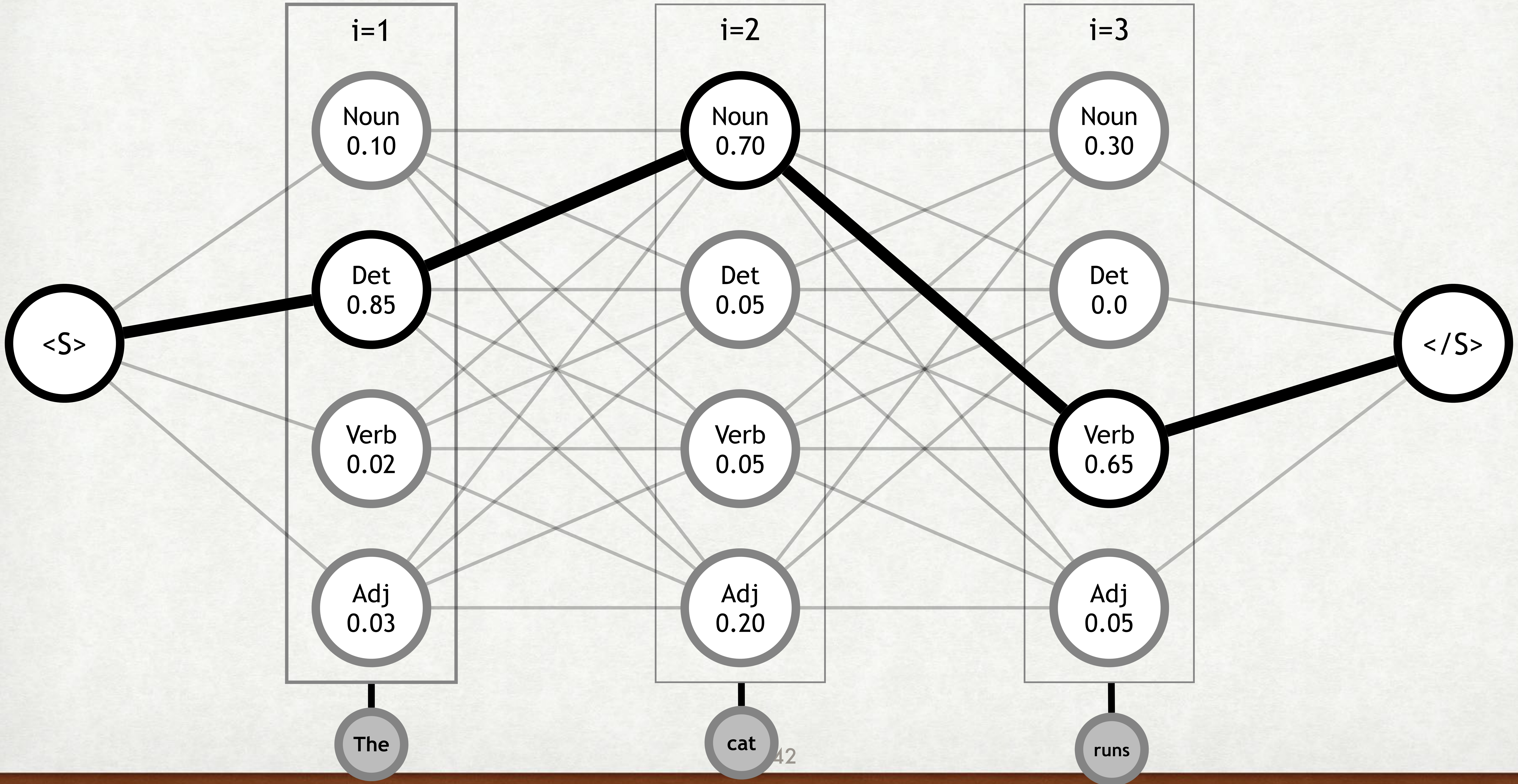
DOWNLOAD THE NOTEBOOK FROM THE CLASS WEBSITE

Break-Out Room Exercise [around 20 minutes]:
https://docs.google.com/document/d/1ifTqeqmK6cG2Zk-f5kDMvU_baNh3x-nbxQXz1d549HY/edit?usp=sharing

TRAINING & DECODING OF CRF: VITERBI/FORWARD BACKWARD ALGORITHM

TRAINING

$$P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)} = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{Z(X)}$$



INTERACTIONS

- each label depends on the input and nearby labels

- but, given *adjacent* labels, the others do not matter!

- If we knew the score of every sequence $y_1, y_2, \ldots y_{n-1}$
  we could easily compute the score of every sequence $y_1, y_2, \ldots y_{n-1}, y_n$

- So, we only really need to know the score of all the sequences ending in *each* y_{n-1}

(Think of that as a *"pre-calculation"* that happens before we think about y_n

STEP 1: INITIAL

First, calculate transition from <S> and emission of the first word for every POS

natural

0:<S> → 1:NN score[“1 NN”] = T(NN|<S>) + S(natural | NN)

0:<S> → 1:JJ score[“1 JJ”] = T(JJ|<S>) + S(natural | JJ)

0:<S> → 1:VB score[“1 VB”] = T(VB|<S>) + S(natural | VB)

0:<S> → 1:LRB score[“1 LRB”] = T(LRB|<S>) + S(natural | LRB)

0:<S> → 1:RRB score[“1 RRB”] = T(RRB|<S>) + S(natural | RRB)

…

STEPS: MIDDLE PARTS

For middle words, calculate the scores for all possible previous POS tags

natural	language
1:NN	2:NN
1:JJ	2:JJ
1:VB	2:VB
1:LRB	2:LRB
1:RRB	2:RRB
…	…

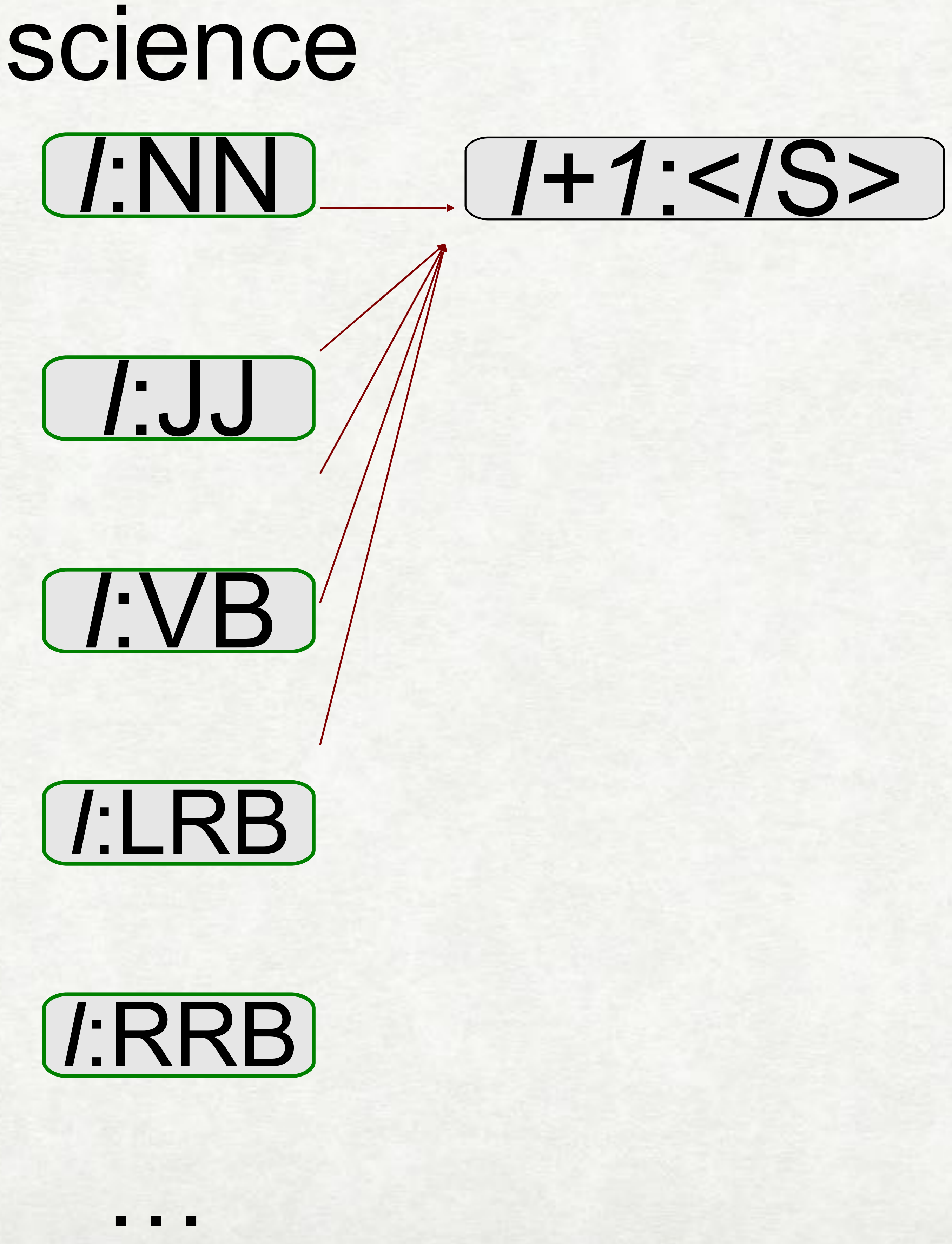
score["2 NN"] = log_sum_exp(
score["1 NN"] + T(NN|NN) + S(language | NN),
score["1 JJ"] + T(NN|JJ) + S(language | NN),
score["1 VB"] + T(NN|VB) + S(language | NN),
score["1 LRB"] + T(NN|LRB) + S(language | NN),
score["1 RRB"] + T(NN|RRB) + S(language | NN),
...)

score["2 JJ"] = log_sum_exp(
score["1 NN"] + T(JJ|NN) + S(language | JJ),
score["1 JJ"] + T(JJ|JJ) + S(language | JJ),
score["1 VB"] + T(JJ|VB) + S(language | JJ),
...

$$log\ sum\ exp(x, y) = \log(\exp(x) + \exp(y))$$

STEPS: FINAL PART

Finish up the sentence with the sentence final symbol



score["*I+1* </S>"] = log_sum_exp(
  score["*I* NN"] + T(</S>|NN),
  score["*I* JJ"] + T(</S>|JJ),
  score["*I* VB"] + T(</S>|VB),
  score["*I* LRB"] + T(</S>|LRB),
  score["*I* NN"] + T(</S>|RRB),
  ...
)