

CS11-747 Neural Networks for NLP

# Structured Prediction with Local Dependencies

Graham Neubig



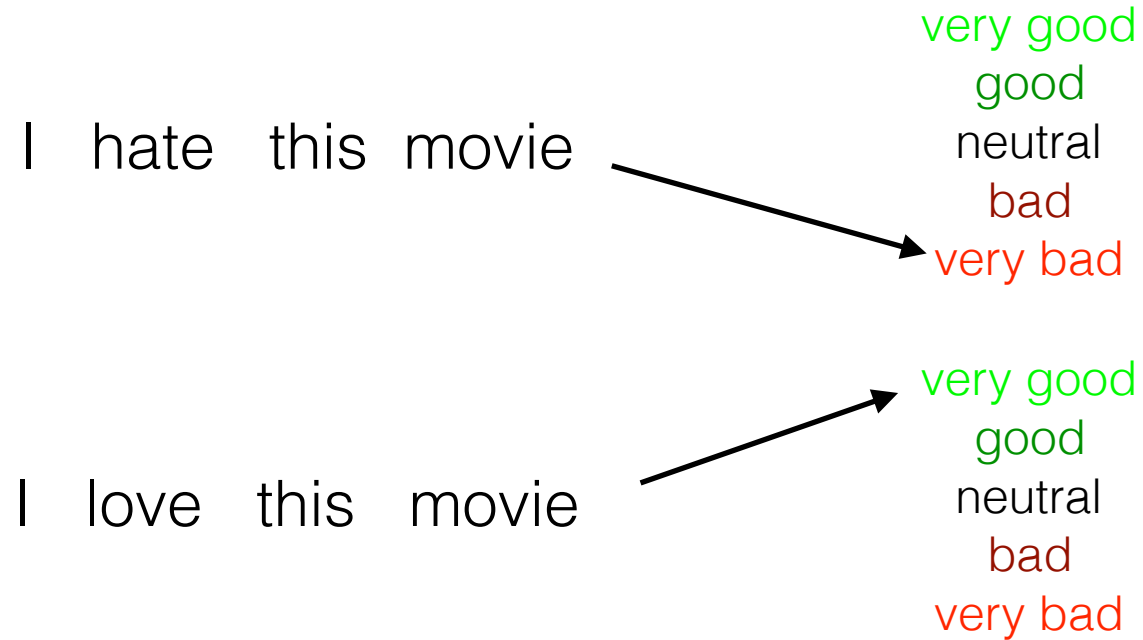
**Carnegie Mellon University**

Language Technologies Institute

<https://phontron.com/class/nn4nlp2020/>


With Slides by Xuezhe Ma

# A Prediction Problem

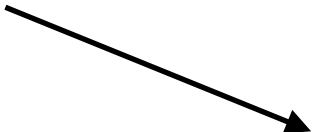


# Types of Prediction


- 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:**

- Maximum likelihood w/ teacher forcing
- Sequence level likelihood w/ dynamic programs
- Reinforcement learning/minimum risk training
- Structured perceptron, structured large margin
- Sampling corruptions of data

Covered  
already

Covered  
today

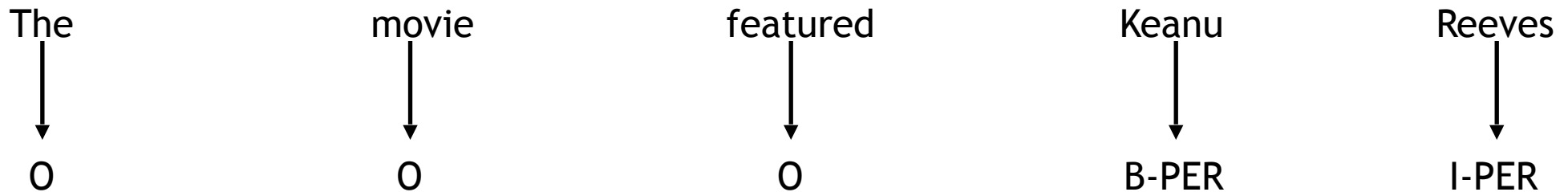
An Example Structured Prediction Problem:  
Sequence Labeling

# Sequence Labeling

- One tag for one word
- e.g. Part of speech tagging



- e.g. Named entity recognition



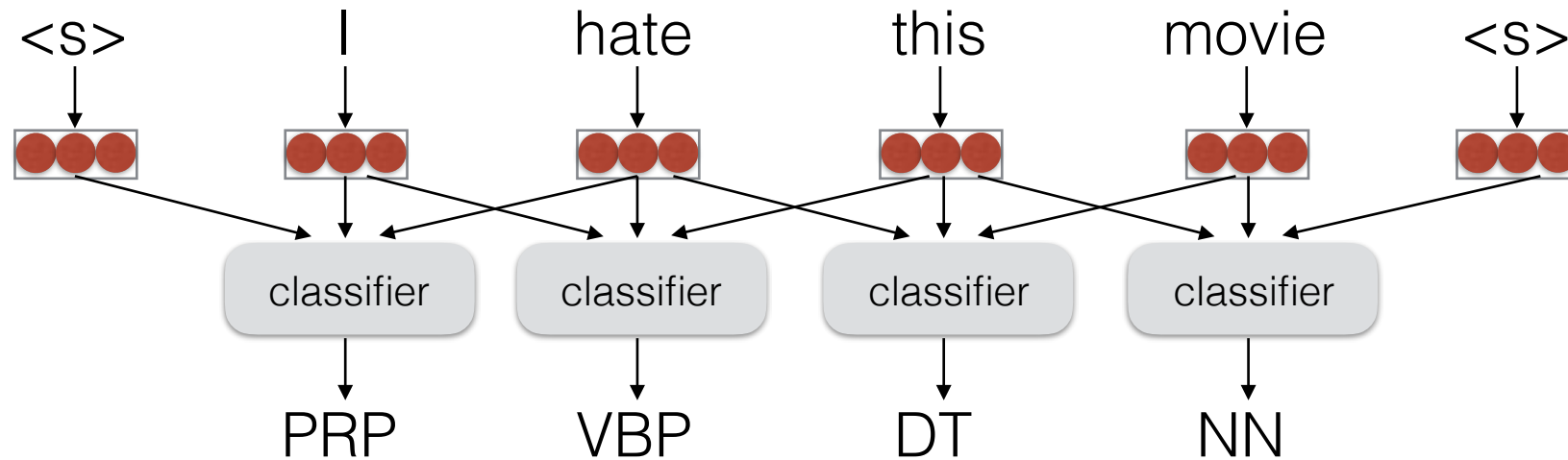
# Why Model Interactions in Output?

- Consistency is important!

time	flies	like	an	arrow	
NN	VBZ	IN	DT	NN	(time moves similarly to an arrow)
NN	NNS	VB	DT	NN	("time flies" are fond of arrows)
VB	NNS	IN	DT	NN	(please measure the time of flies similarly to how an arrow would)
		↓	max frequency		
NN	NNS	IN	DT	NN	("time flies" that are similar to an arrow)

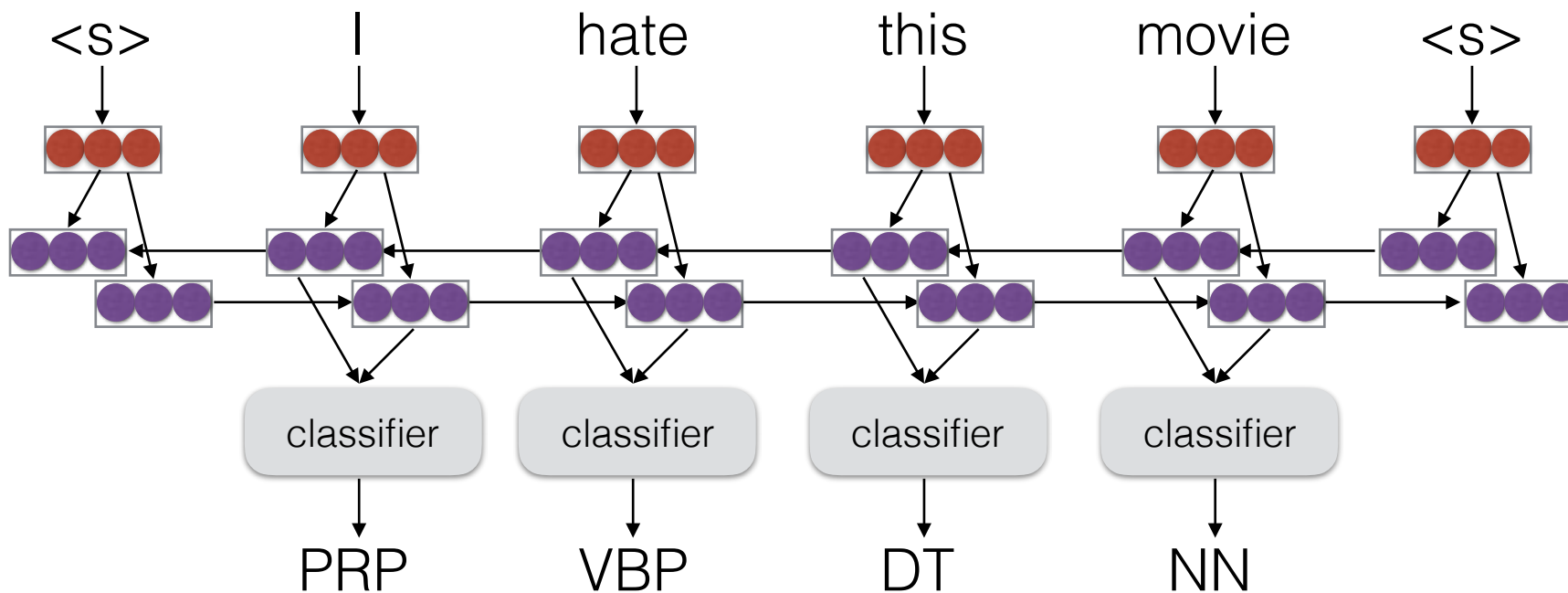


# Sequence Labeling as Independent Classification



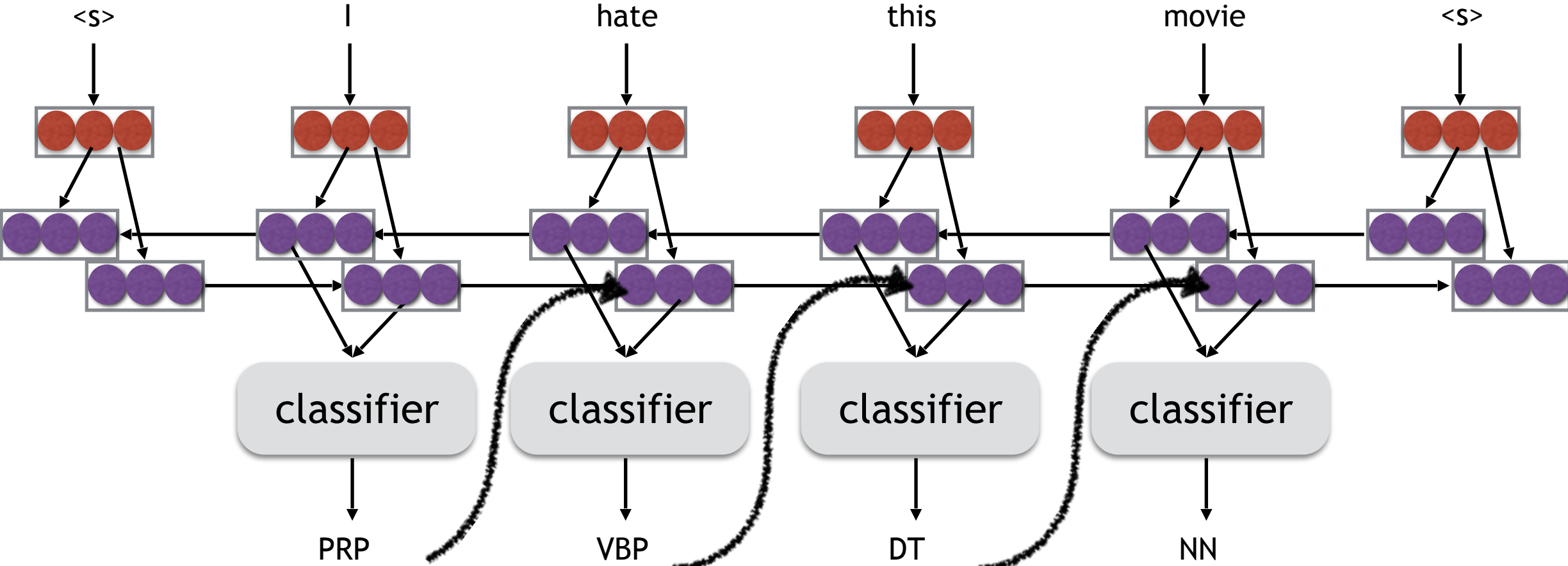
- Structured prediction task, but not structured prediction model: multi-class classification

# Sequence Labeling w/ BiLSTM



- Still not modeling output structure! Outputs are independent

# Recurrent Decoder



# Problems

Independent classification models:

- Strong independence assumptions
- No guarantee of valid or consistent structures

History-based/sequence-to-sequence models

- No independence assumptions
- Cannot calculate exactly! Require approximate search
- Exposure bias

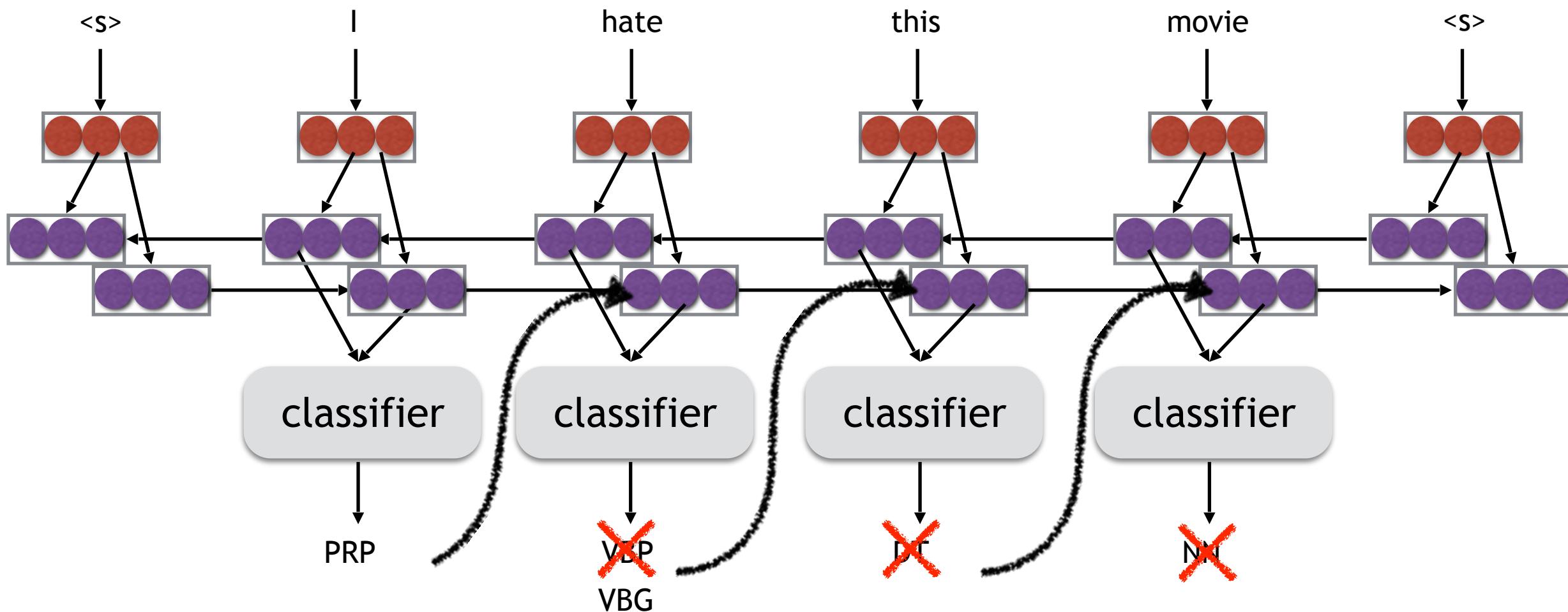
# Teacher Forcing and Exposure Bias

**Teacher Forcing:** The system is trained receiving only the correct inputs during training.

**Exposure Bias:** At inference time, it receives the previous predictions, which could be wrong!

→ The model has never been "exposed" to these errors, and fails.

# An Example of Exposure Bias



Models w/ Local Dependencies:  
**Conditional Random Fields**

# Models w/ Local Dependencies

- Some independence assumptions on the output space, but not entirely independent (local dependencies)
- Exact and optimal decoding/training via dynamic programs

**Conditional Random Fields!**  
**(CRFs)**



# Local Normalization vs. Global Normalization

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

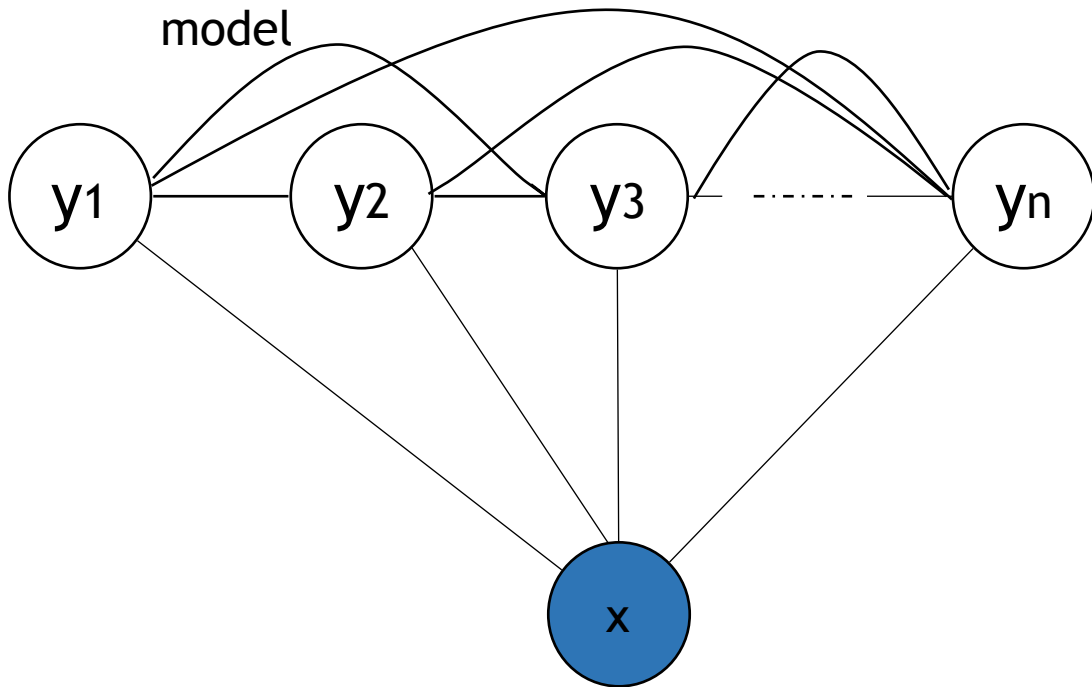
$$P(Y | X) = \prod_{j=1}^{|Y|} \frac{e^{S(y_j | X, y_1, \dots, y_{j-1})}}{\sum_{\tilde{y}_j \in V} e^{S(\tilde{y}_j | X, y_1, \dots, y_{j-1})}}$$

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

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

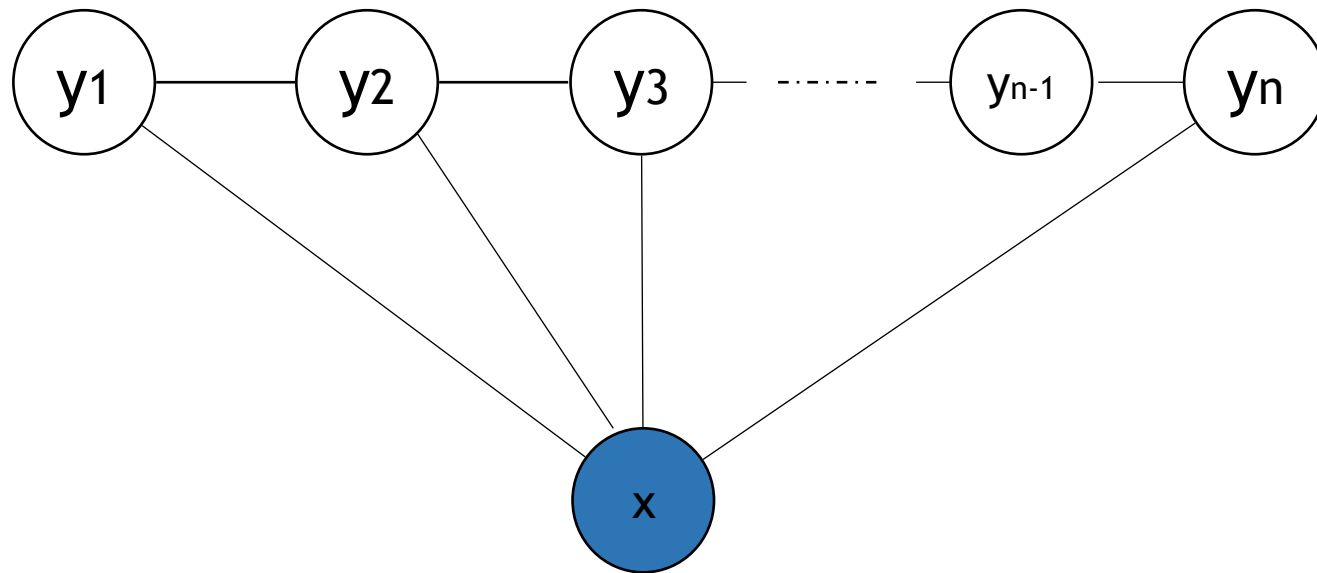
# Conditional Random Fields

General form of globally normalized model

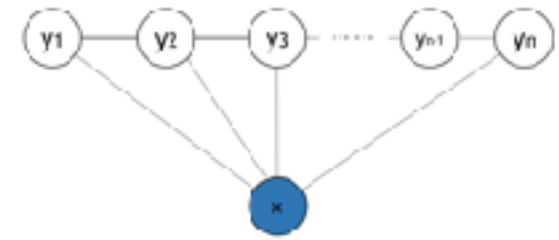


$$P(Y|X) = \frac{\psi(Y, X)}{\sum_{Y'} \psi(Y', X)}$$

First-order linear CRF



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



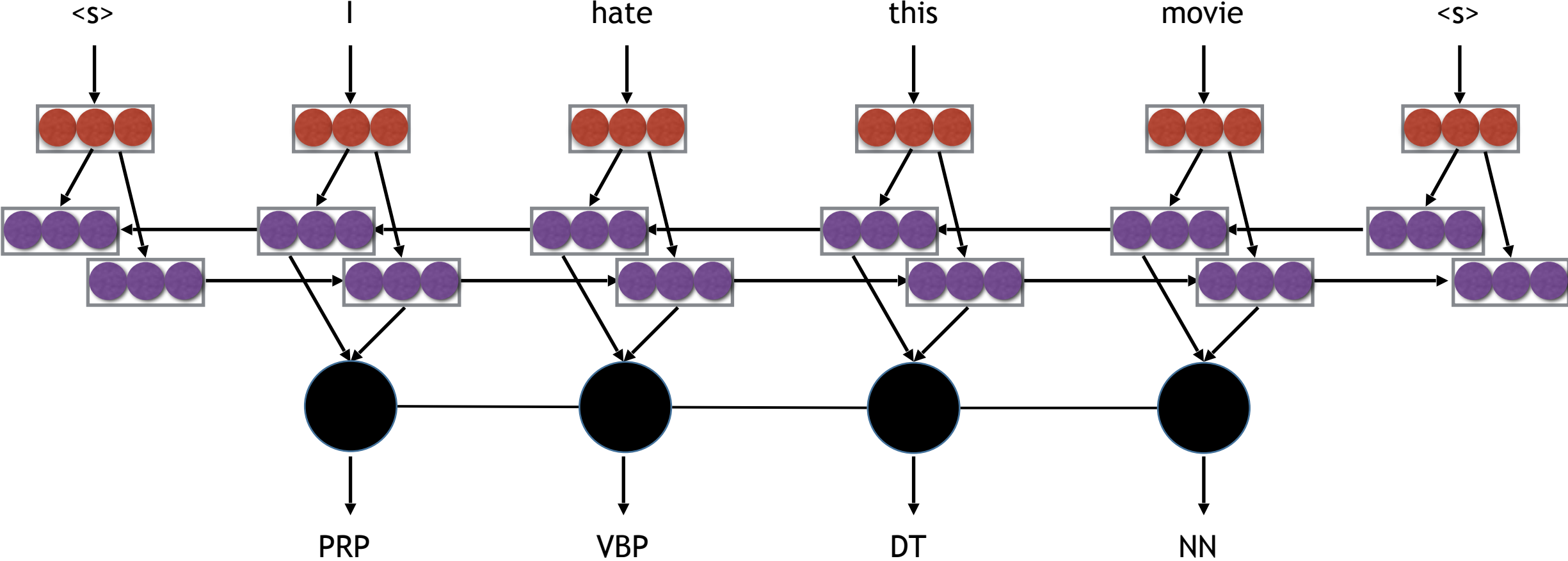
# Potential Functions

"Transition"

"Emission"

- $\psi_i(y_{i-1}, y_i, X) = \exp(\boxed{W^T T(y_{i-1}, y_i, X, i)} + \boxed{U^T S(y_i, X, i)} + b_{y_{i-1}, y_i})$

# BiLSTM-CRF for Sequence Labeling



# Training & Decoding of CRF: Viterbi/Forward Backward Algorithm

# CRF Training & Decoding

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

Easy to compute

Hard to compute

Go through the output space of Y which grows exponentially with the length of the input sequence.

# Interactions

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

# Forward Calculation: Initial Part

- First, calculate transition from  $\langle S \rangle$  and emission of the first word for every POS





# Forward Calculation 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"] +  $\Psi(\text{NN}, \text{NN})$  +  $\Psi(y_1 = \text{NN}, X)$ ,  
score["1 JJ"] +  $\Psi(\text{JJ}, \text{NN})$  +  $\Psi(y_1 = \text{NN}, X)$ ,  
score["1 VB"] +  $\Psi(\text{VB}, \text{NN})$  +  $\Psi(y_1 = \text{NN}, X)$ ,  
score["1 LRB"] +  $\Psi(\text{LRB}, \text{NN})$  +  $\Psi(y_1 = \text{NN}, X)$ ,  
score["1 RRB"] +  $\Psi(\text{RRB}, \text{NN})$  +  $\Psi(y_1 = \text{NN}, X)$ ,  
...)

score["2 JJ"] = log\_sum\_exp(  
score["1 NN"] +  $\Psi(\text{NN}, \text{JJ})$  +  $\Psi(y_1 = \text{JJ}, X)$ ,  
score["1 JJ"] +  $\Psi(\text{JJ}, \text{JJ})$  +  $\Psi(y_1 = \text{JJ}, X)$ ,  
score["1 VB"] +  $\Psi(\text{VB}, \text{JJ})$  +  $\Psi(y_1 = \text{JJ}, X)$ ,  
...)

...

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

# Forward Calculation: Final Part

- Finish up the sentence with the sentence final symbol

science

L:NN → L+1:</S>

L:JJ

L:VB

L:LRB

L:RRB

...

```
score["L+1 </S>"] = log_sum_exp(  
  score["L NN"] + Ψ(NN,</S>),  
  score["L JJ"] + Ψ(JJ,</S>),  
  score["L VB"] + Ψ(VB,</S>),  
  score["L LRB"] + Ψ(LRB,</S>),  
  score["L RRB"] + Ψ(RRB,</S>),  
  ...  
)
```

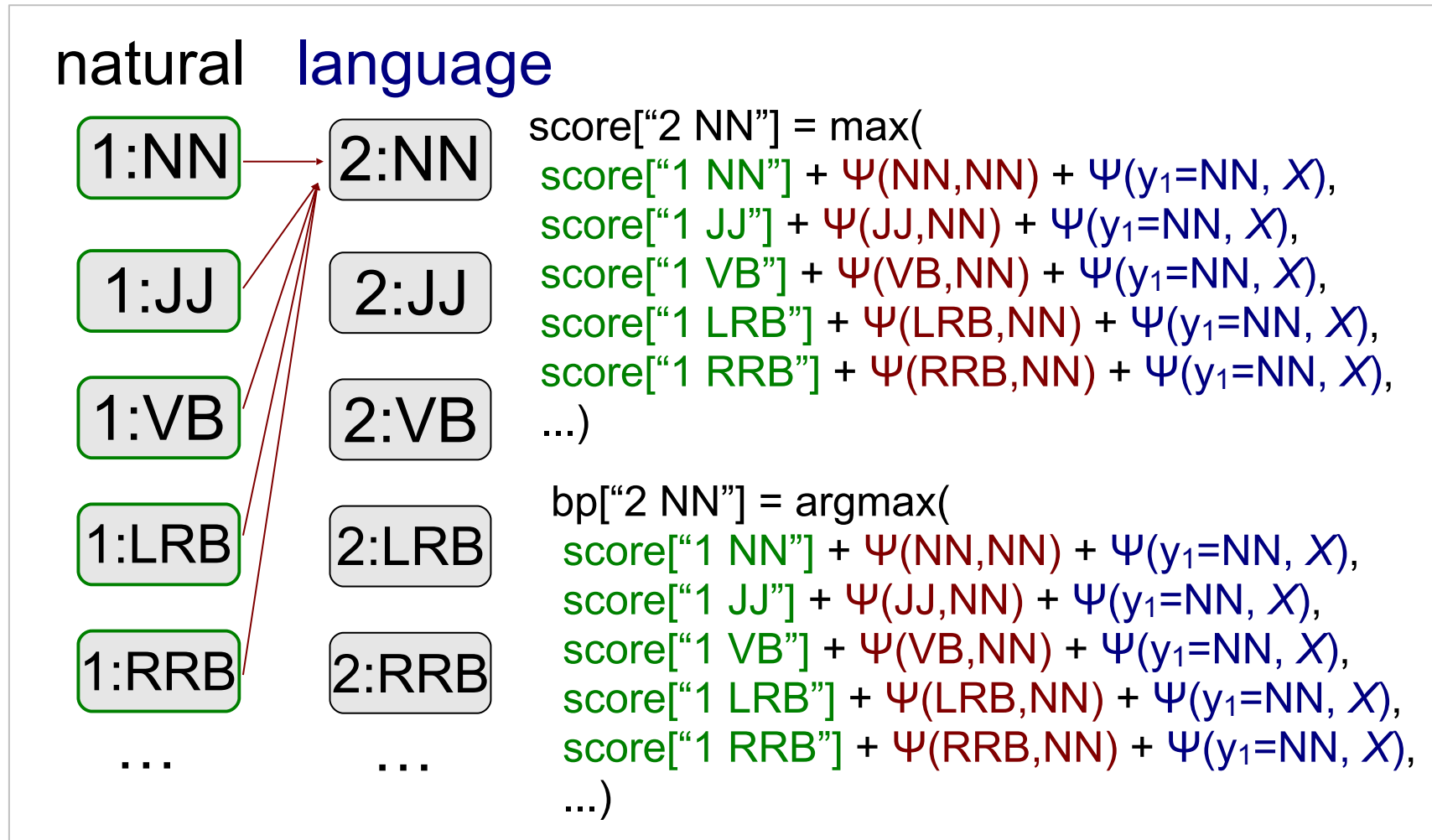
# Revisiting the Partition Function

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

- **Cumulative score** of " $\langle S \rangle$ " at position  $L+1$  now is the sum of all paths, equal to partition function  $Z(X)$ !
- Subtract this from (log) score of true path to calculate global log likelihood to use as **loss function**.
- ("**backward**" step in traditional CRFs handled by our neural net/ autograd toolkit.)

# Argmax Search

- **Forward step:** Instead of `log_sum_exp`, use "max", maintain back-pointers



- **Backward step:** Re-trace back-pointers from end to beginning

# Case Study

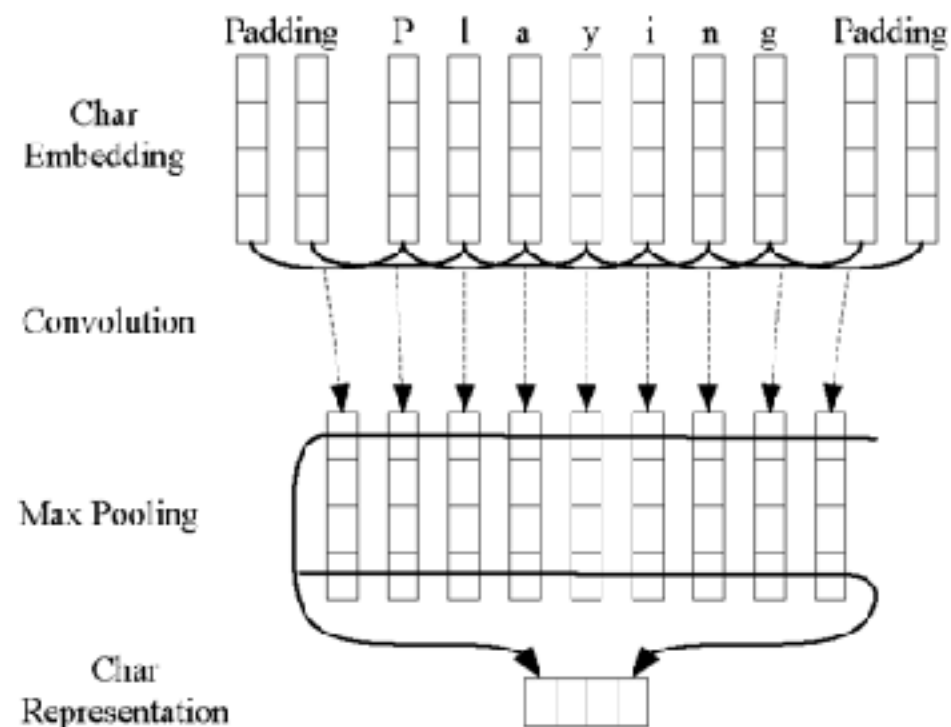
BiLSTM-CNN-CRF for Sequence Labeling

# Case Study: BiLSTM-CNN-CRF for Sequence Labeling (Ma et al, 2016)

- **Goal:** Build end-to-end neural model for sequence labeling, requiring no feature engineering and data pre-processing.
- Two levels of representations
  - Character-level representation: CNN
  - Word-level representation: Bi-directional LSTM

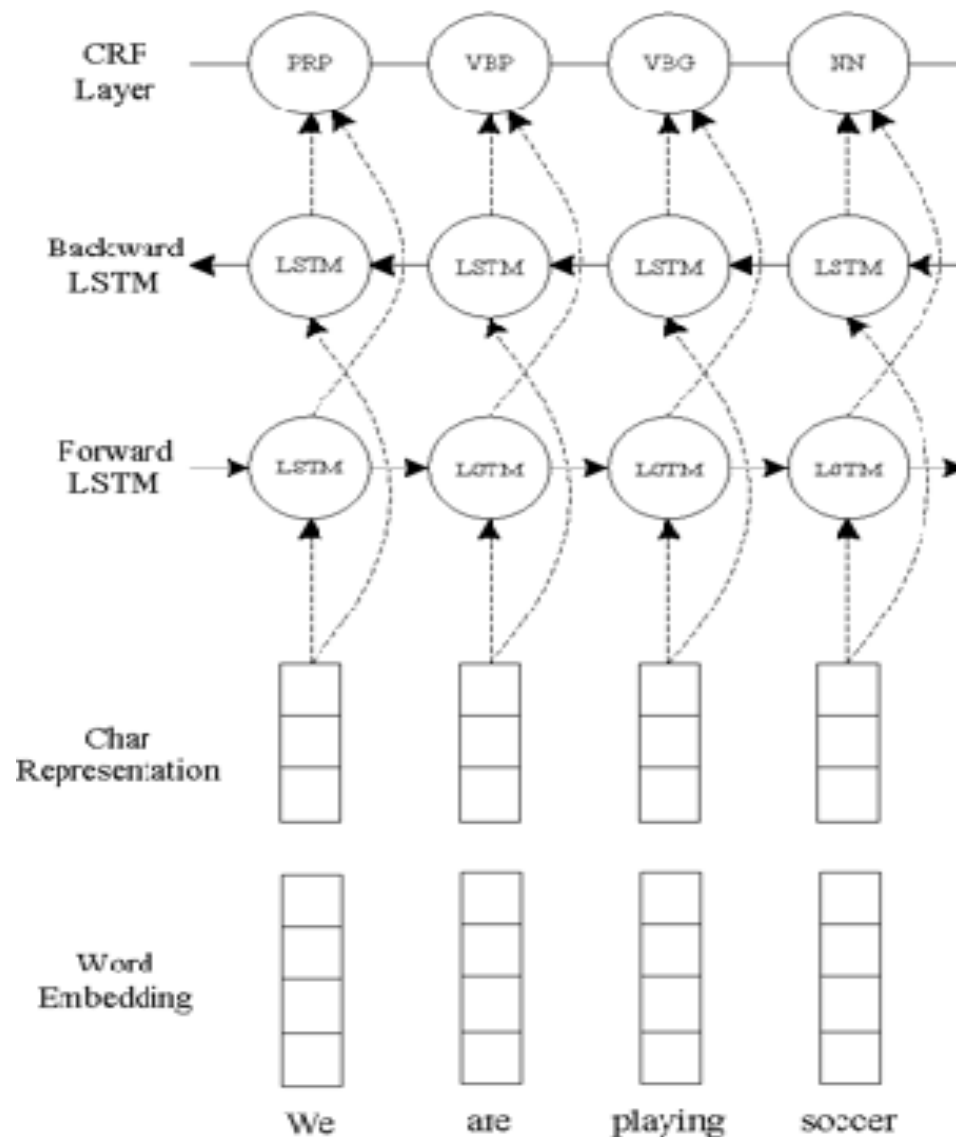
# CNN for Character-level representation

- CNN to extract morphological information such as prefix or suffix of a word



# Bi-LSTM-CNN-CRF

- Bi-LSTM to model word-level information.
- CRF is on top of Bi-LSTM to consider the correlation between labels.





# Training Details

- Optimization Algorithm:
  - SGD with momentum (0.9)
  - Learning rate decays with rate 0.05 after each epoch.
- Dropout Training:
  - Applying dropout to regularize the model with fixed dropout rate 0.5
- Parameter Initialization:
  - Parameters: Glorot and Bengio (2010)
  - Word Embedding: Stanford's GloVe 100-dimensional embeddings
  - Character Embedding: uniformly sampled from  $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$ , where  $dim = 30$

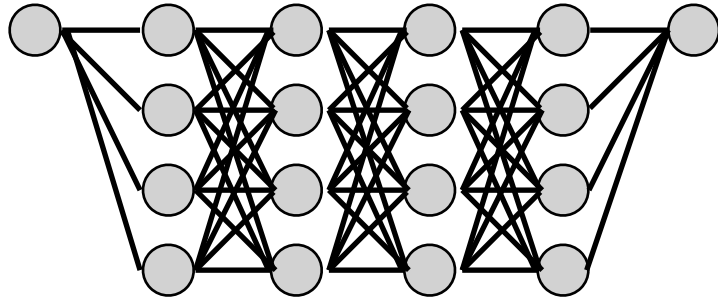
# Experiments

<b>Model</b>	<b>POS</b>		<b>NER</b>					
	<b>Dev</b>	<b>Test</b>	<b>Dev</b>			<b>Test</b>		
	<b>Acc.</b>	<b>Acc.</b>	<b>Prec.</b>	<b>Recall</b>	<b>F1</b>	<b>Prec.</b>	<b>Recall</b>	<b>F1</b>
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BLSTM-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

# Generalized CRFs

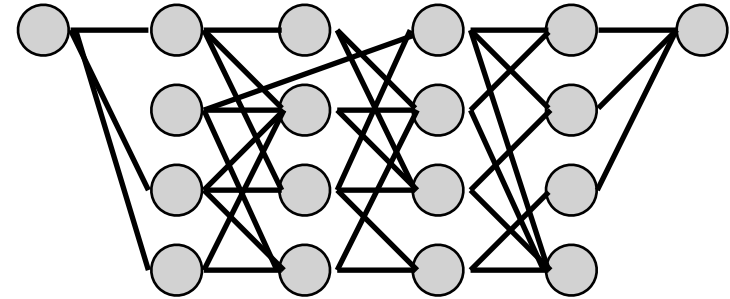
# Data Structures to Marginalize Over

## Fully Connected Lattice/Trellis



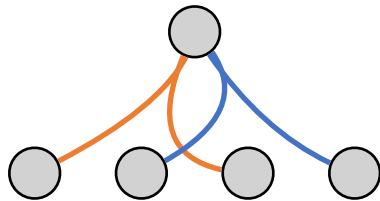
(this is what a linear-chain CRF looks like)

## Sparsely Connected Lattice/Graph



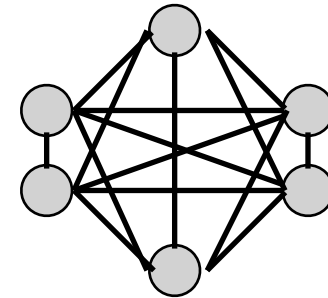
(e.g. speech recognition lattice, trees)

## Hyper-graphs



(for example, multiple tree candidates)

## Fully Connected Graph



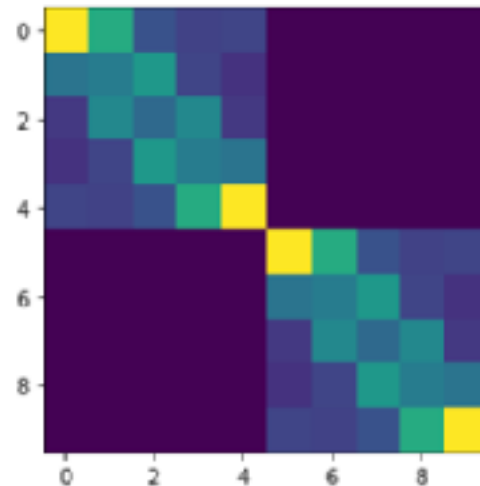
(e.g. full seq2seq models, dynamic programming not possible)

# Generalized Dynamic Programming Models

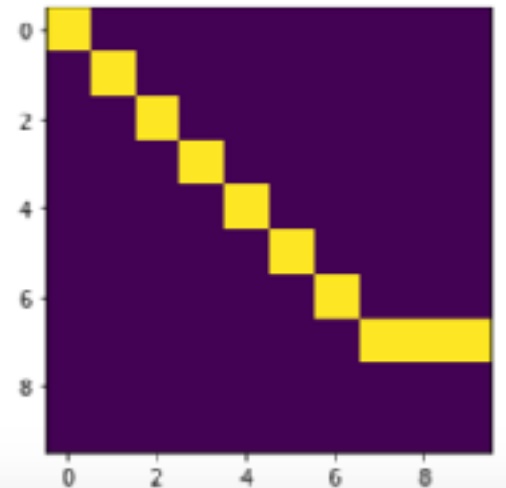
- **Decomposition Structure:** What structure to use, and thus also what dynamic programming to perform?
- **Featurization:** How do we calculate local scores?
- **Score Combination:** How do we combine together scores? e.g. `log_sum_exp`, `max` (concept of "semi-ring")
- Example: `pytorch-struct`

<https://github.com/harvardnlp/pytorch-struct>

```
# Compute marginals  
show(dist.marginals[0])
```



```
# Compute argmax  
show(dist.argmax.detach()[0])
```



Questions?