CS11-747 Neural Networks for NLP

Structured Prediction with Local Dependencies

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

https://phontron.com/class/nn4nlp2018/

With Many Slides by Xuezhe Ma

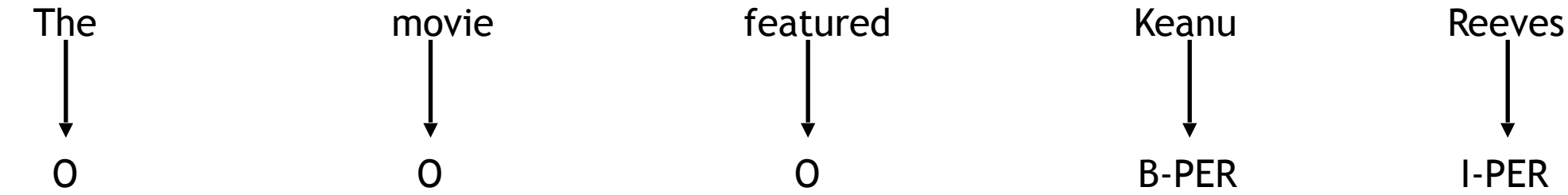
An Example Structured Prediction Problem:

Sequence Labeling

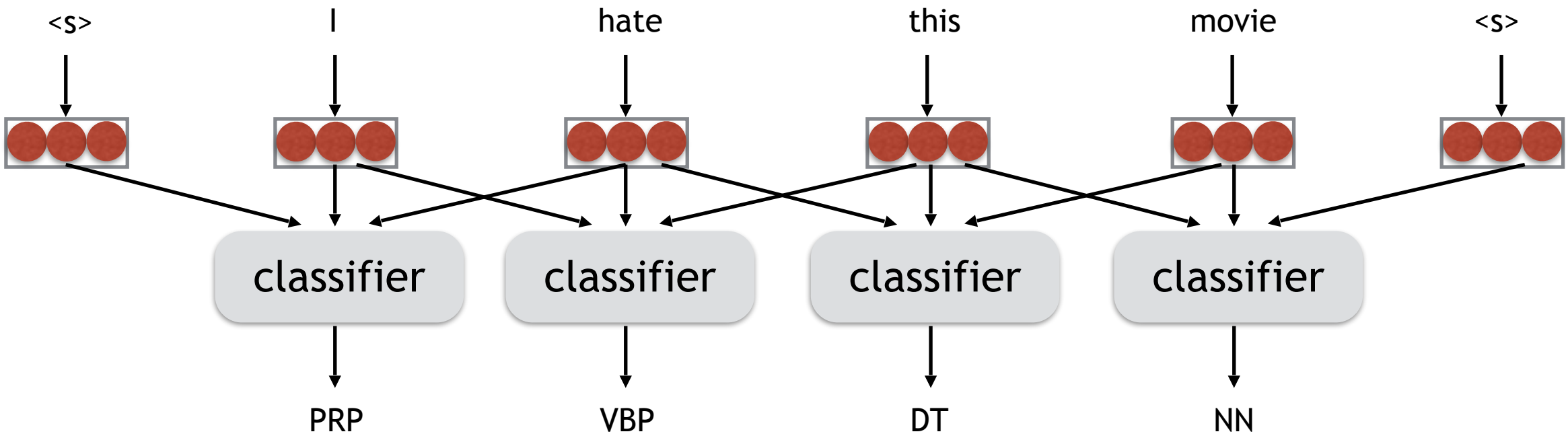
Sequence Labeling

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

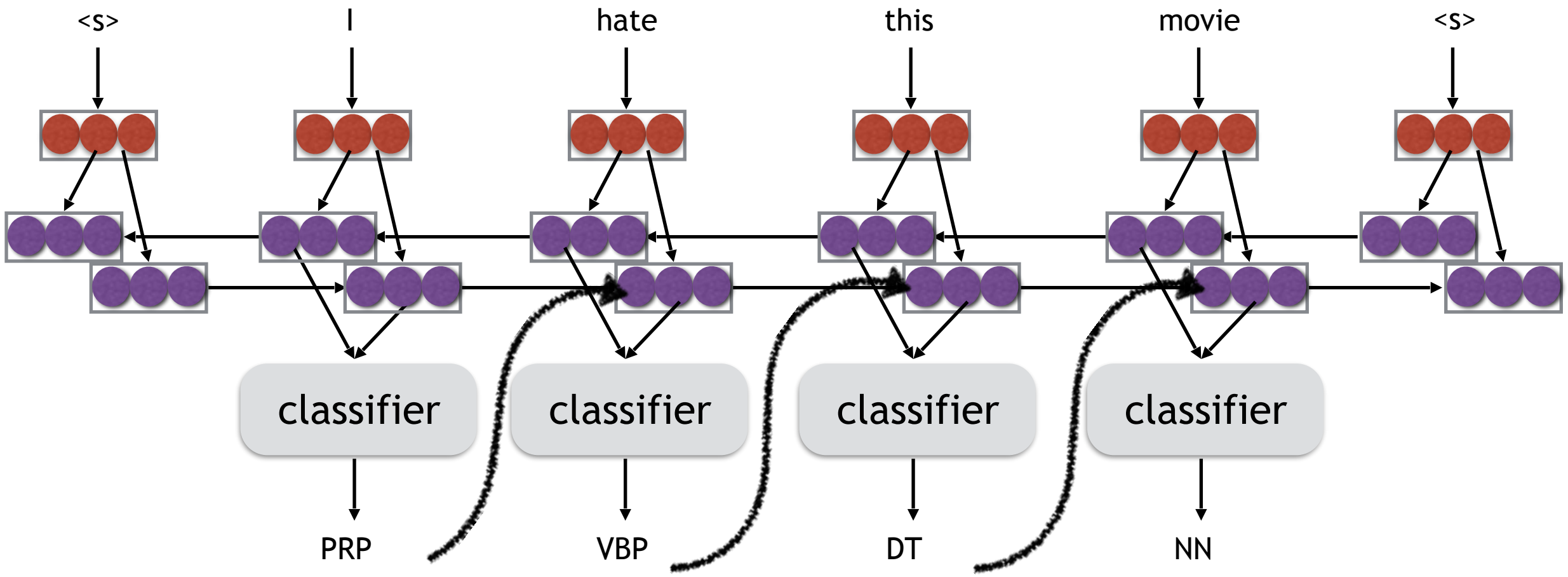
- e.g. Named entity recognition



Sequence Labeling as Independent Classification



Teacher Forcing



Previously Covered Models

- 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, 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)

Reminder: Globally Normalized Models

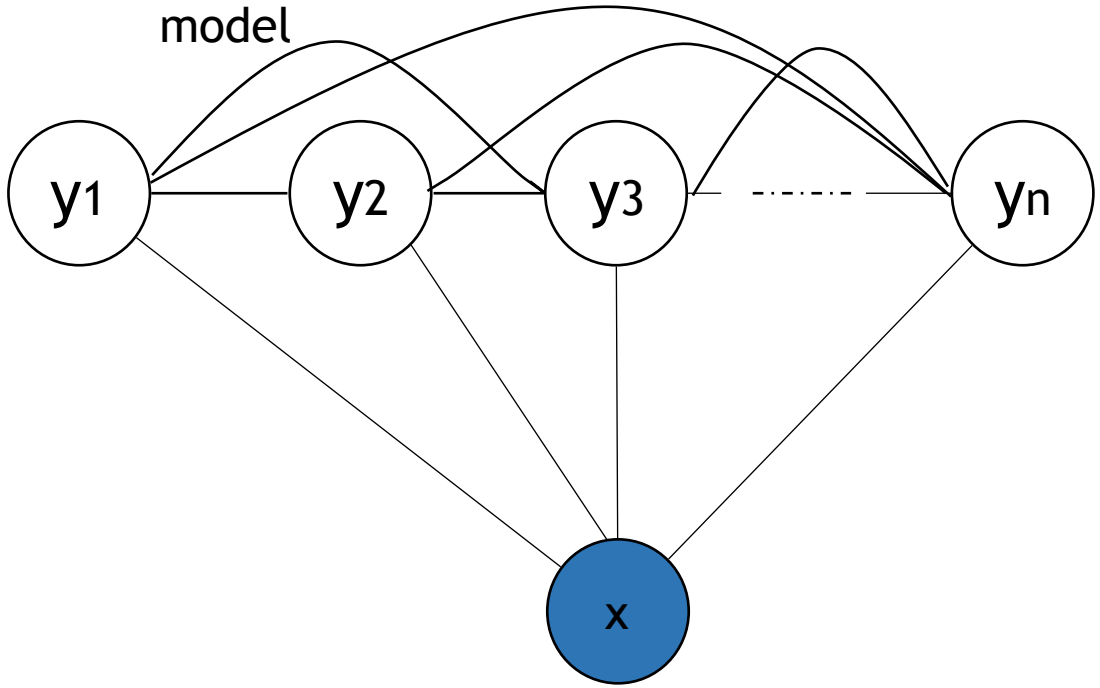
- Each output sequence has a score, which is not normalized over a particular decision

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

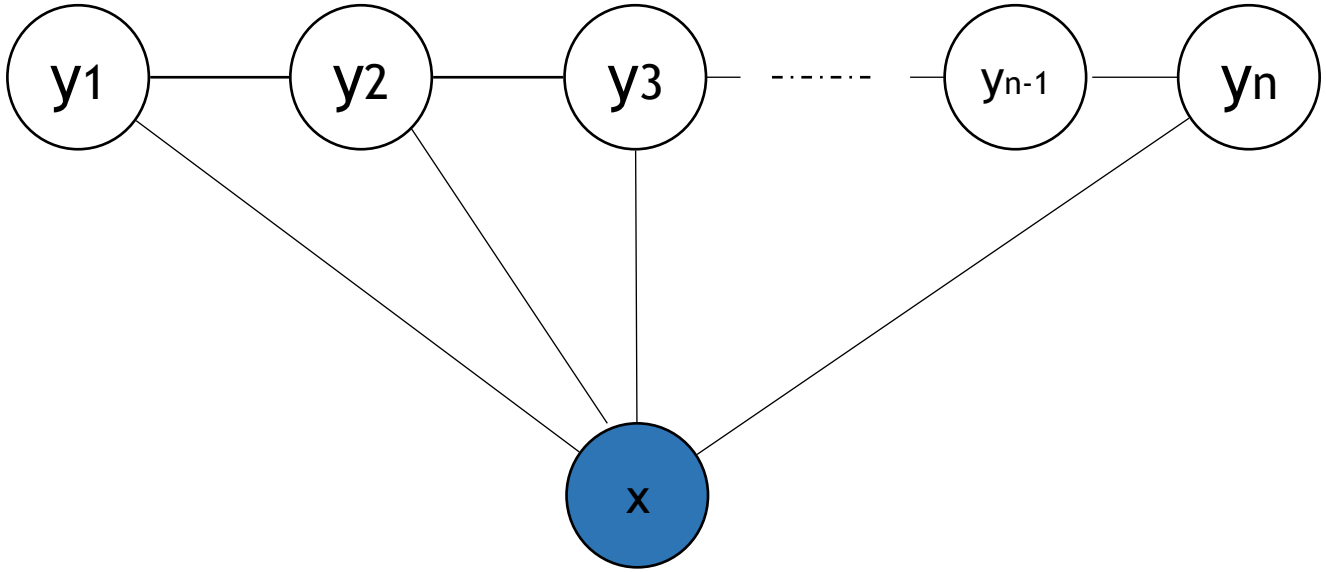
where $\psi(Y,X)$ are potential functions.

Conditional Random Fields

General form of globally normalized model



First-order linear CRF



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

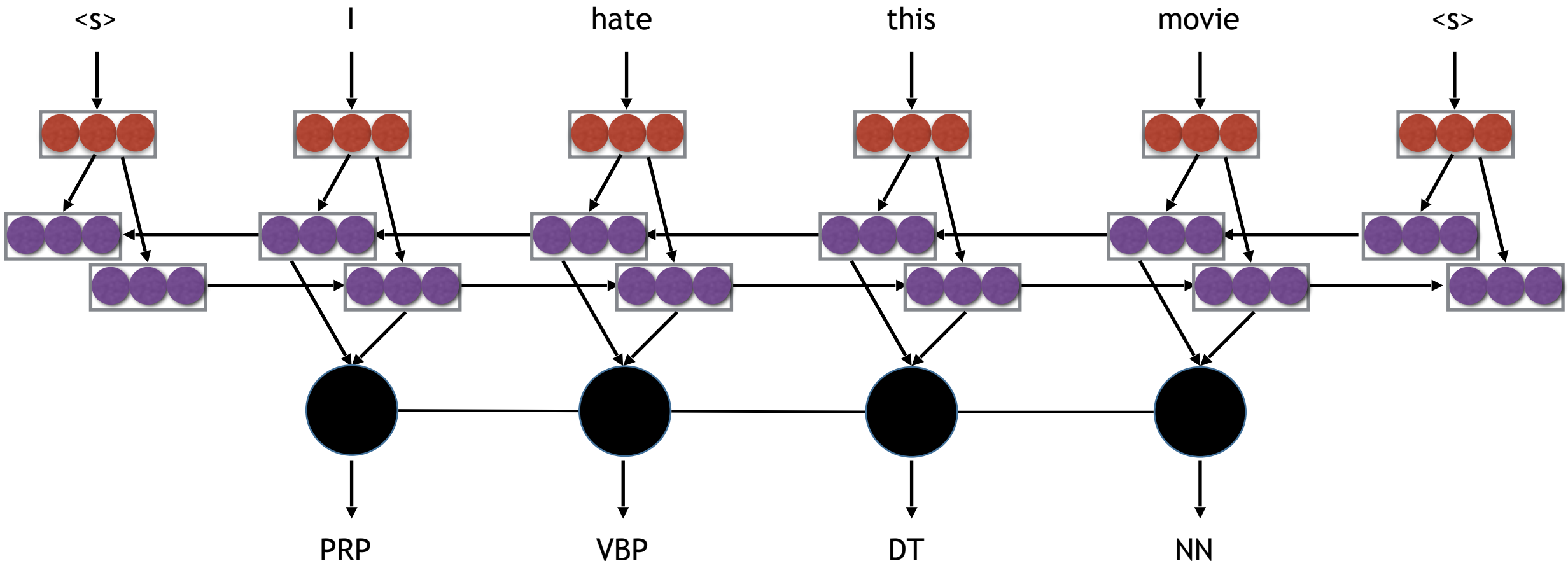
$$P(Y|X) = \frac{\prod_{i=1}^{L}\psi_i(y_{i-1},y_i,X)}{\sum_{Y'}\prod_{i=1}^{L}\psi_i(y'_{i-1},y'_i,X)}$$

Potential Functions

"Transition" "Emission"

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

BiLSTM-CRF for Sequence Labeling



Training & Decoding of CRF: Viterbi/Forward Backward Algorithm

CRF Training & Decoding

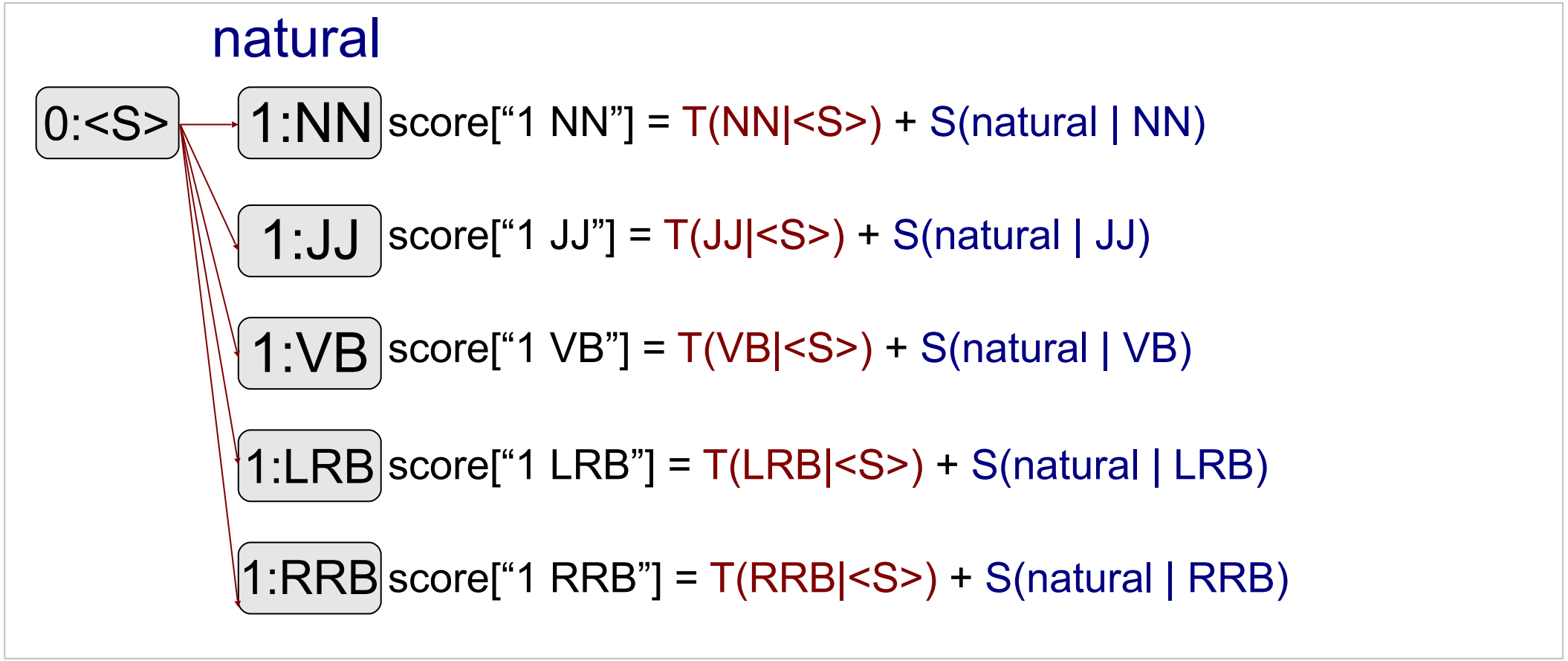
- $P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)} = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{Z(X)}$

Interactions

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

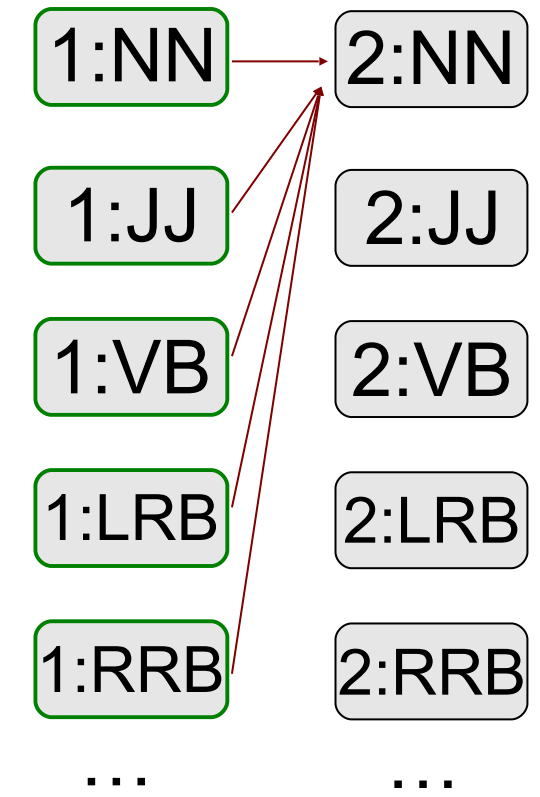
Step: Initial Part

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



Step: Middle Parts

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



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),
...)

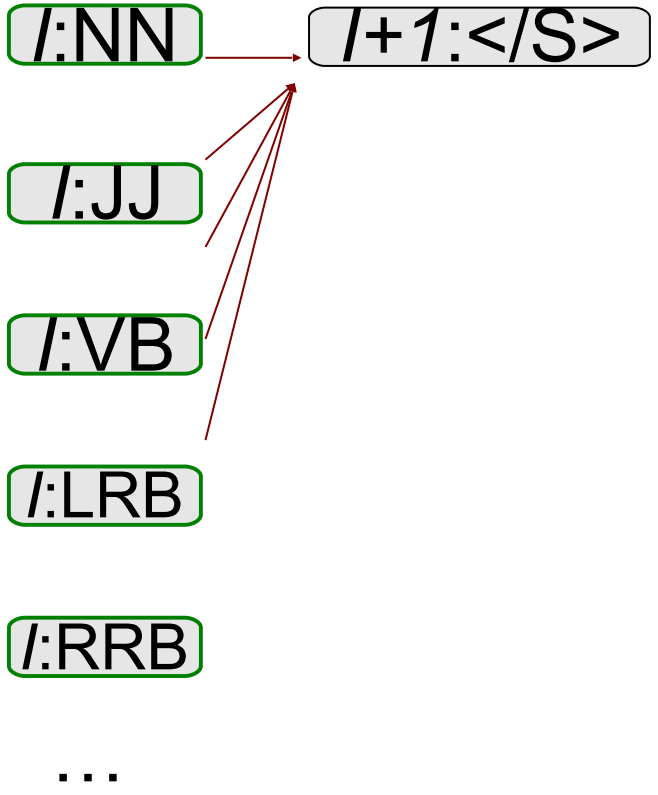
score[“2 JJ”] = log_sum_exp(
score[“1 NN”] + T(JJ|NN) + S(language | JJ),
score[“1 JJ”] + T(JJ|JJ) + S(language | JJ),
score[“1 VB”] + T(JJ|VB) + S(language | JJ),
...

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

Forward Step: Final Part

- Finish up the sentence with the sentence final symbol

science



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),
...
)

Computing the Partition Function

- $\pi_t(y|X)$ is the partition of sequence with length equal to t and end with label y:

$$\pi_t(y|X) = \sum_{y_i,\cdots,y_{t-1}} \left(\prod_{i=1}^{t-1} \psi_i(y_{i-1}, y_i, X) \right) \psi_t(y_{t-1}, y_t = y, X)$$

$$= \sum_{y_{t-1}} \psi_t(y_{t-1}, y_t = y, X) \sum_{y_i,\cdots,y_{t-2}} \left(\prod_{i=1}^{t-2} \psi_i(y_{i-1}, y_i, X) \right) \psi_{t-1}(y_{t-2}, y_{t-1}, X)$$

$$= \sum_{y_{t-1}} \psi_t(y_{t-1}, y_t = y, X) \pi_{t-1}(y_{t-1}|X)$$

- Computing partition function $Z(X) = \sum_y \pi_L(y|X)$

Decoding and Gradient Calculation

- Decoding is performed with similar dynamic programming algorithm
- Calculating gradient: $l_{ML}(X, Y; \theta) = -\log P(Y|X; \theta)$

$$\frac{\partial l_{ML}(X, Y; \theta)}{\partial \theta} = F(Y, X) - E_{P(Y|X;\theta)}[F(Y, X)]$$

  - Forward-backward algorithm (Sutton and McCallum, 2010)
    - Both $P(Y|X; \theta)$ and $F(Y, X)$ can be decomposed
    - Need to compute the marginal distribution:

$$P(y_{i-1} = y', y_i = y|X; \theta) = \frac{\alpha_{i-1}(y'|X)\psi_i(y', y, X)\beta_i(y|X)}{Z(X)}$$

  - Not necessary if using DNN framework (auto-grad)

Case Study

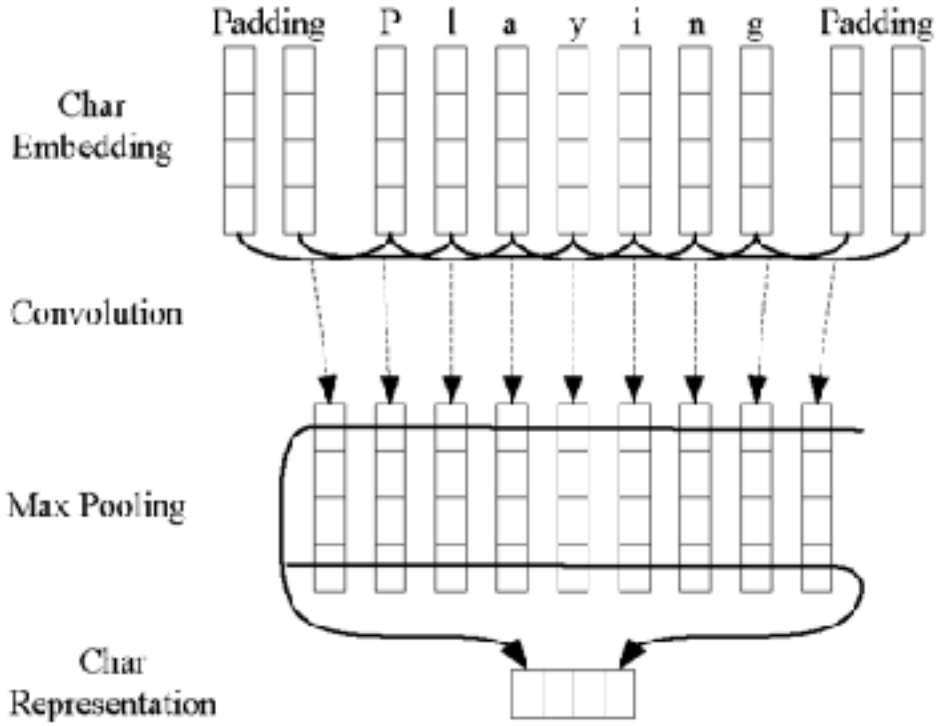
BiLSTM-CNN-CRF for Sequence Labeling

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

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

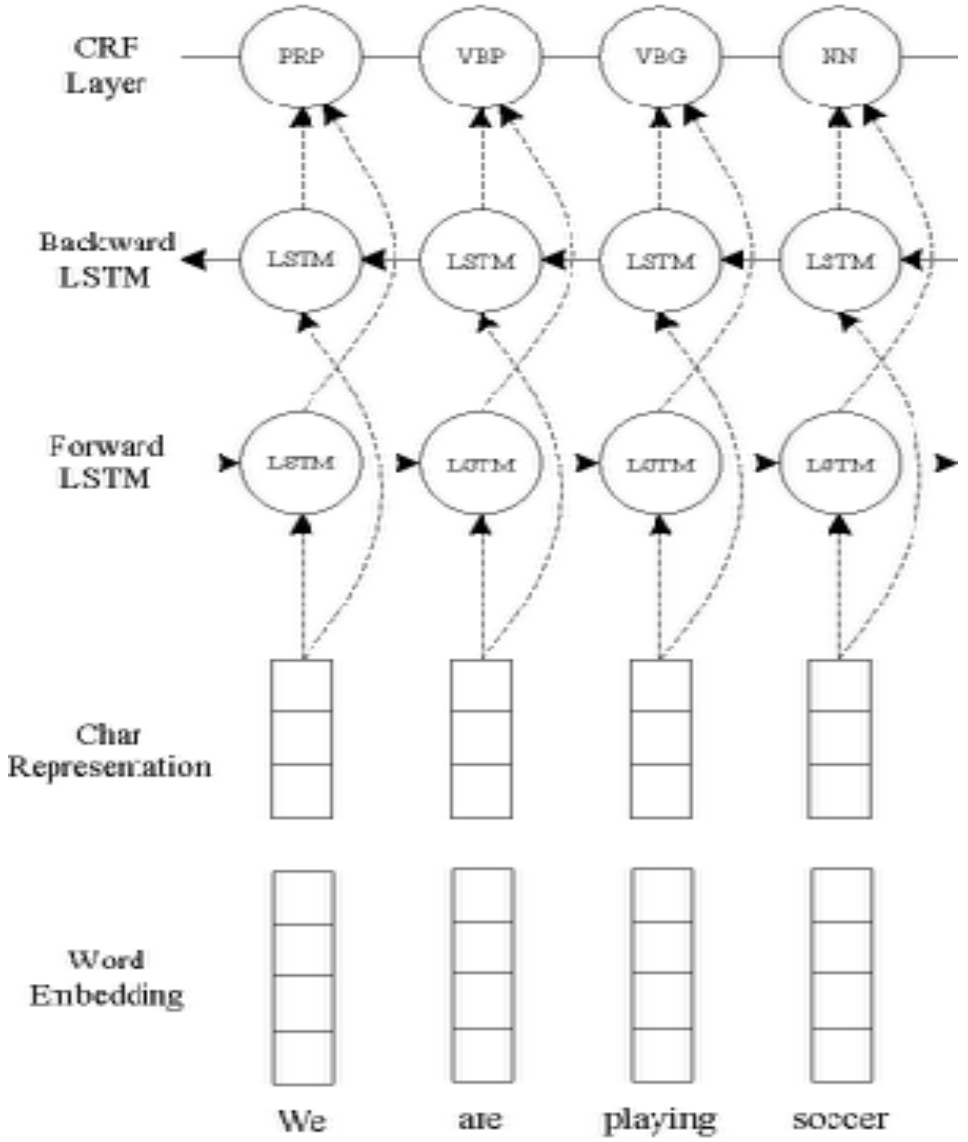
CNN for Character-level representation

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



Bi-LSTM-CNN-CRF

- We used Bi-LSTM to model word-level information.
- CRF is on top of Bi-LSTM to consider the co-relation 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-dimentional embeddings
  - Character Embedding: uniformly sampled from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$, where $dim = 30$

Experiments

Model	POS		NER					
	Dev	Test	Dev			Test		
	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
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

Minimum Risk Training:
Considering Rewards in Globally Normalized Models

Reward Functions in Structured Prediction

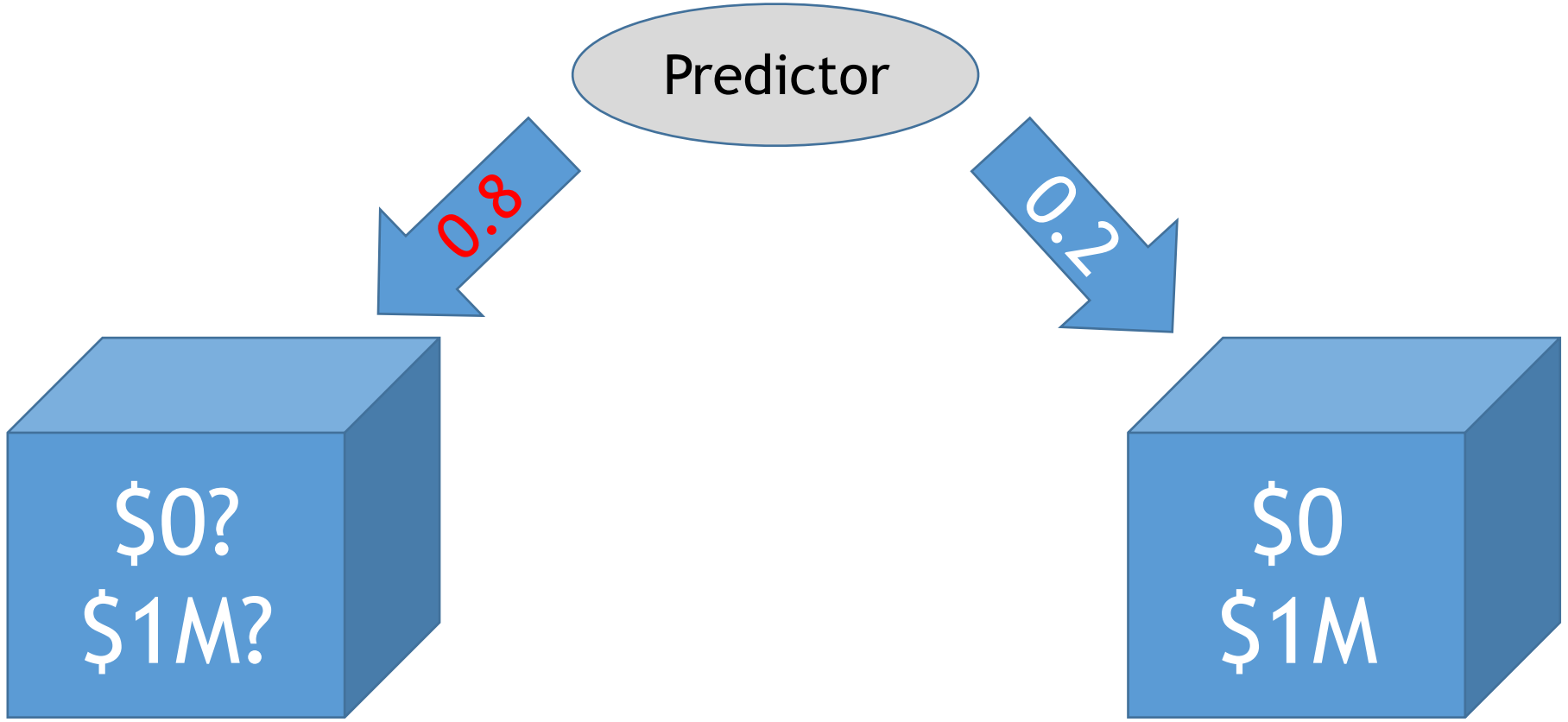
- POS tagging: token-level accuracy
- NER: F1 score
- Dependency parsing: labeled attachment score
- Machine translation: corpus-level BLEU

Do different reward functions impact our decisions?

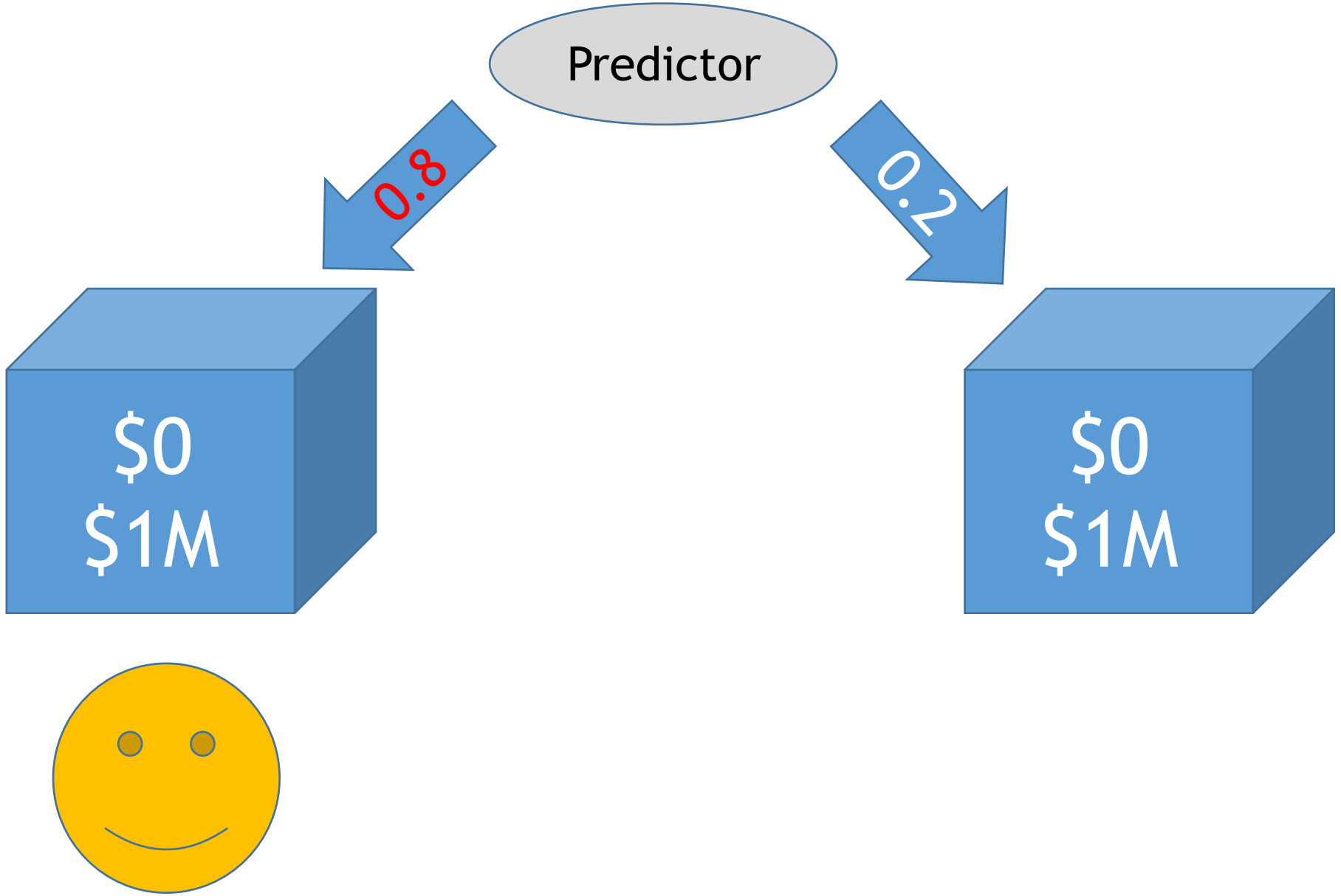
- Data1: $(X, Y) \sim P$
- Task1: predict Y given X i.e. $h_1(X)$
- Reward1: $R_1(h_1(X), Y)$

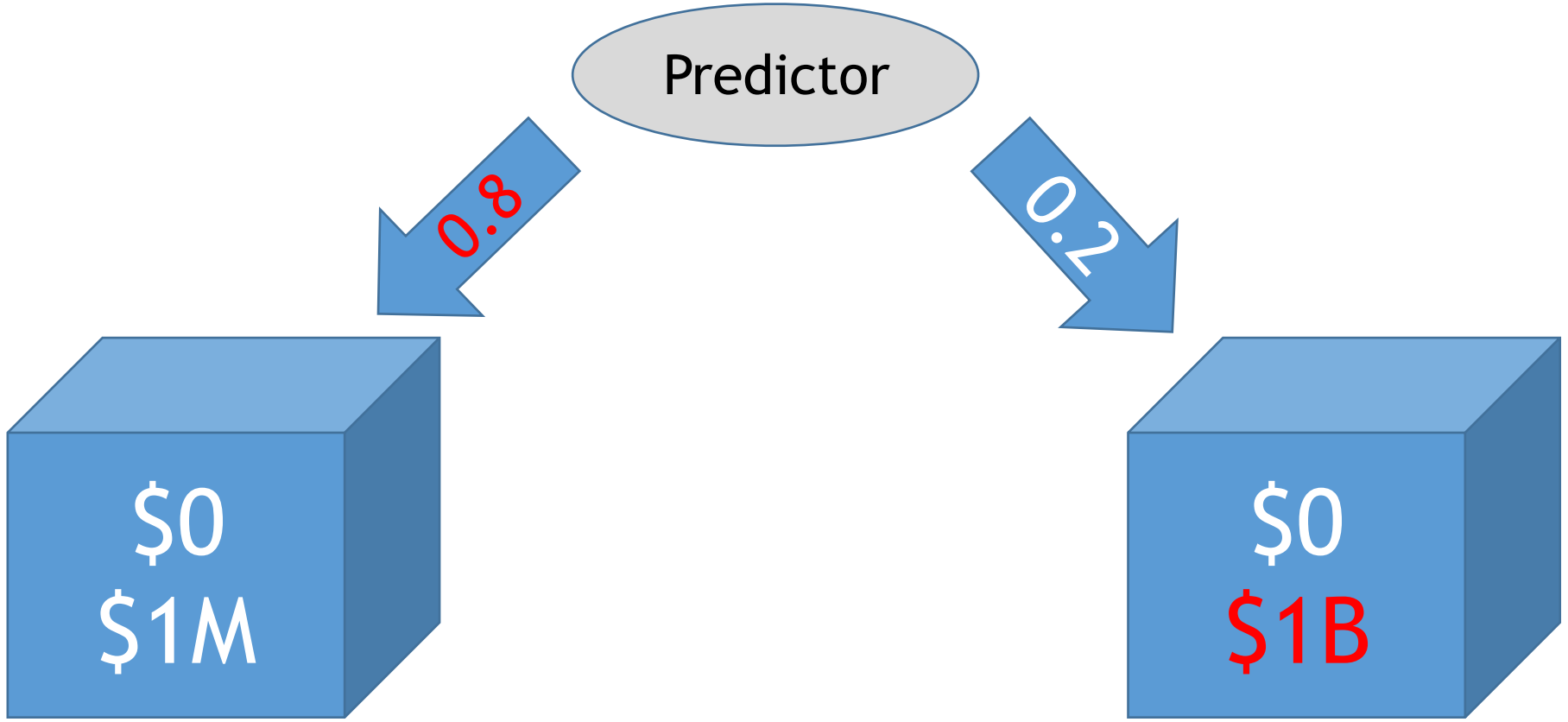
- Data2: $(X, Y) \sim P$
- Task2: predict Y given X i.e. $h_2(X)$
- Reward2: $R_2(h_2(X), Y)$

$$h_1(X) = h_2(X)?$$

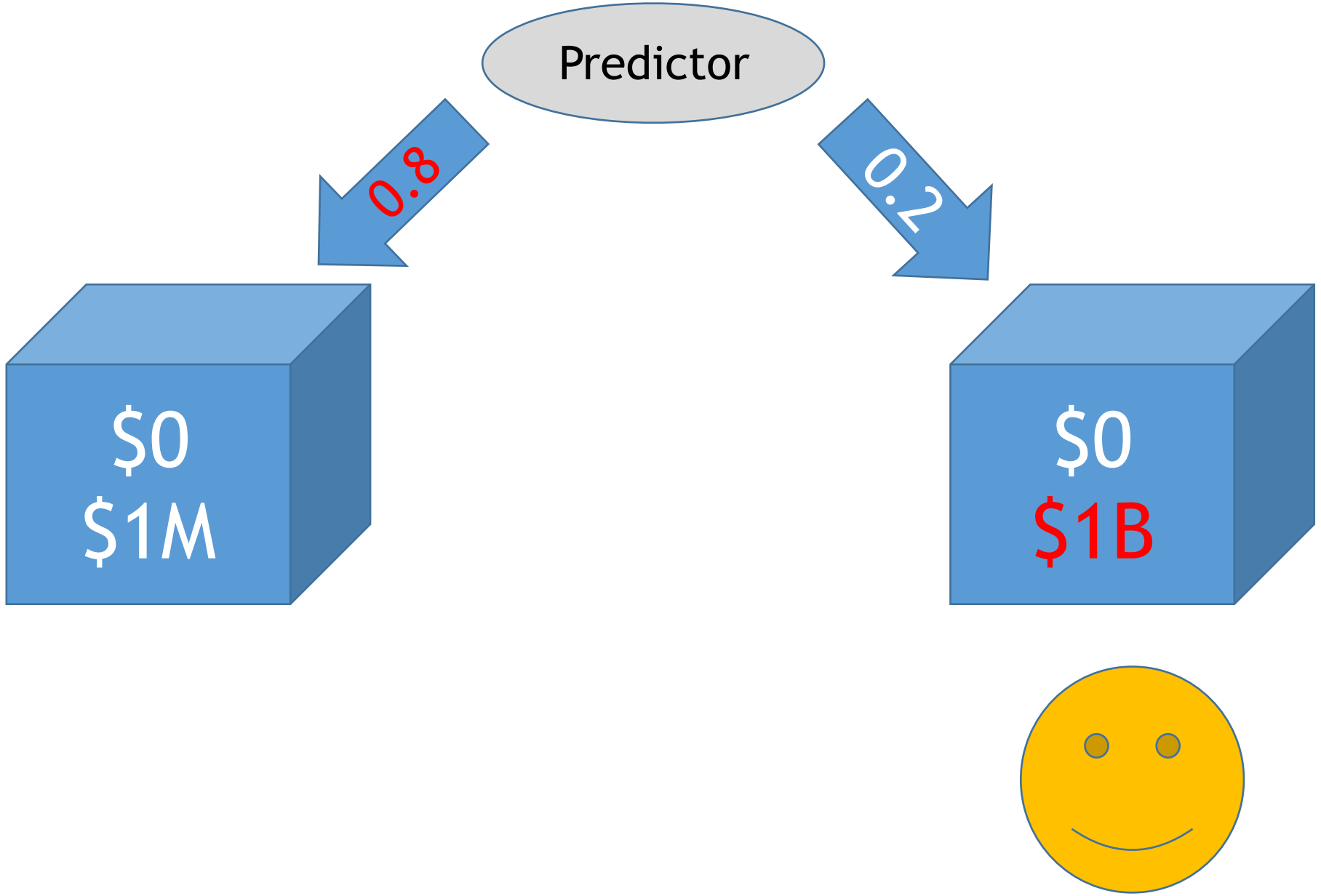


Reward is the amount of money we get


Predictor
0.8
0.2
$0
$1M
$0
$1M



Reward is the amount of money we get


Predictor
0.8
0.2
$0
$1M
$0
$1B

Previous Methods to Consider Reward

- Max-Margin (Taskar et al., 2004)
- REINFORCE (e.g. Ranzato et al. 2016)
- Reward-augmented Maximum Likelihood (Norouzi et al., 2016)

Minimizing Risk by Enumeration

- Simple idea: directly calculate the risk of *all* hypotheses in the space

$$\mathcal{L}_{\text{risk}}(X, Y) = \sum_{Y'} P(Y' \mid X)\text{error}(Y, Y')$$

- Use this as a loss function

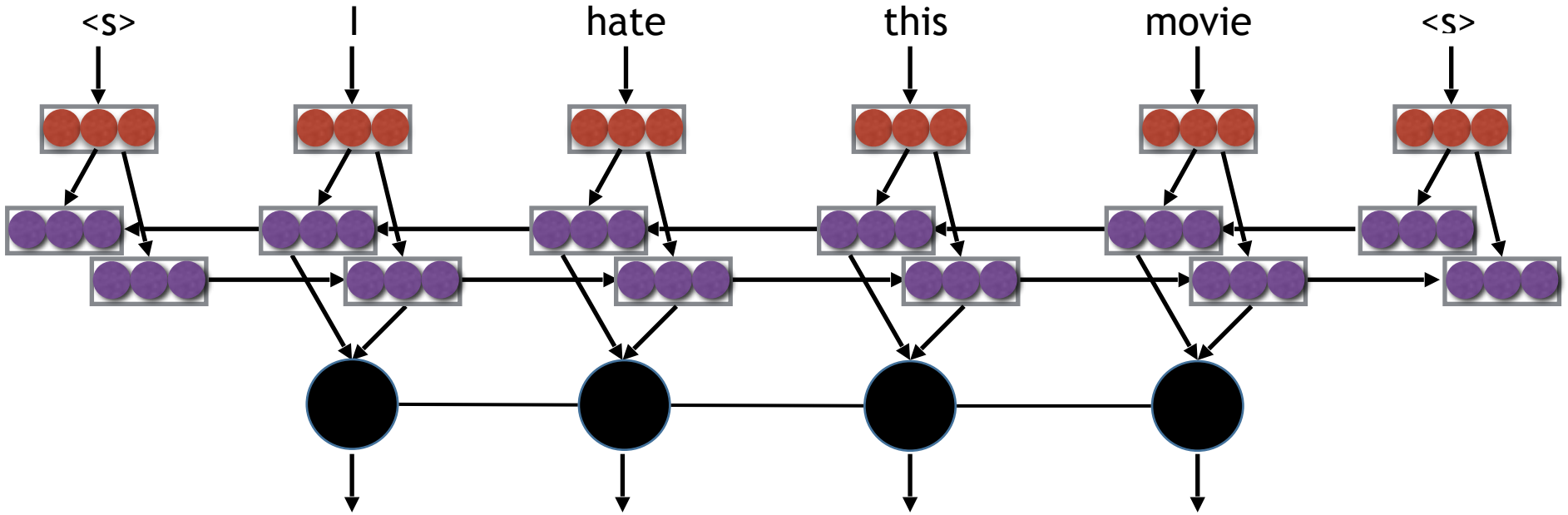
Enumeration + Sampling (Shen+ 2016)

- Enumerating all hypotheses is intractable!
- Instead of enumerating over everything, only enumerate over a sample, and re-normalize

$$\mathcal{L}_{\text{risk}}(X, Y) = \sum_{Y' \in \mathcal{S}} \frac{P(Y' \mid X)}{\sum_{Y'' \in \mathcal{S}} P(Y'' \mid X)} \text{error}(Y, Y')$$

Token-wise Minimum Risk

- If we can come up with a decomposable error function, we can calculate risk for each word



$$\begin{matrix} \text{NN} \\ \text{VBP} \\ \text{PRP} \\ \text{DT} \\ \dots \end{matrix} \begin{bmatrix} 0.3 \\ 0.1 \\ 0.4 \\ 0.05 \\ \dots \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 0 \\ 3 \\ \dots \end{bmatrix}$$

Questions?