CS11-747 Neural Networks for NLP

Structured Prediction with Local Dependencies

Xuezhe Ma (Max)

Carnegie Mellon University

Language Technologies Institute

Site

https://phontron.com/class/nn4nlp2017/

An Example Structured Prediction Problem:
Sequence Labeling

Sequence Labeling

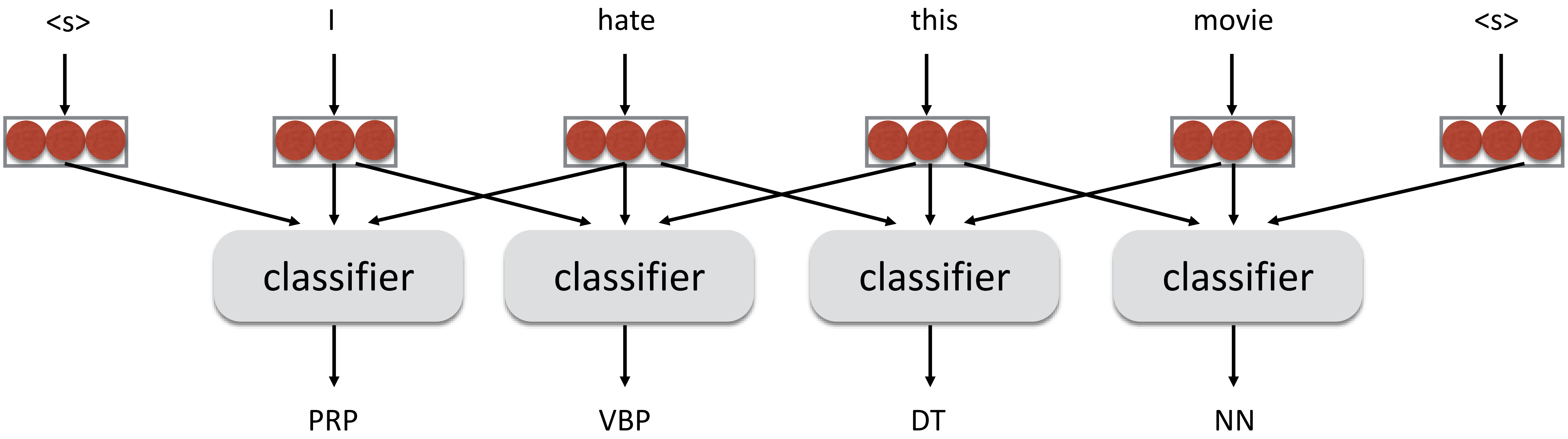
- One tag for one word
- e.g. Part of speech tagging

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

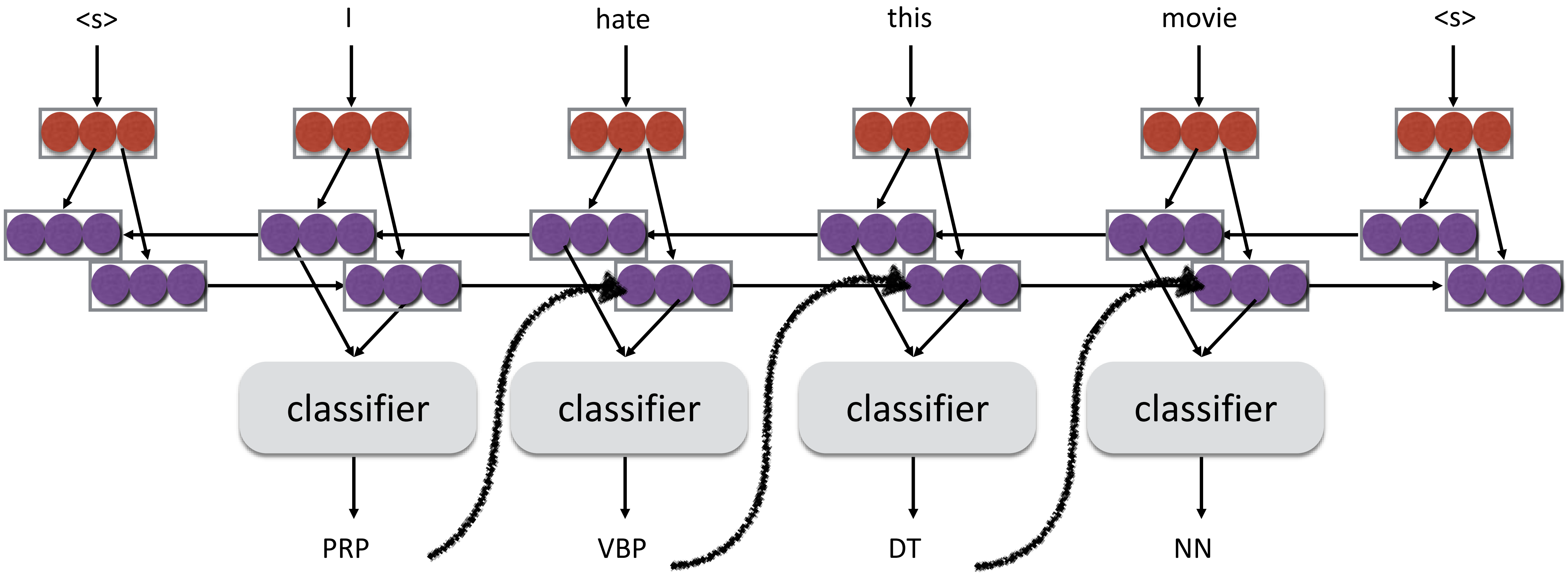
- e.g. Named entity recognition

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

Sequence Labeling as Independent Classification



Locally Normalized Models



Summary

- 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

Globally normalized models?

- Not too strong independent assumption (local dependencies)
- Optimal decoding

Globally normalized models?

- Not too strong independent assumption (local dependencies)
- Optimal decoding

Conditional Random Fields (CRFs)

Globally Normalized Models

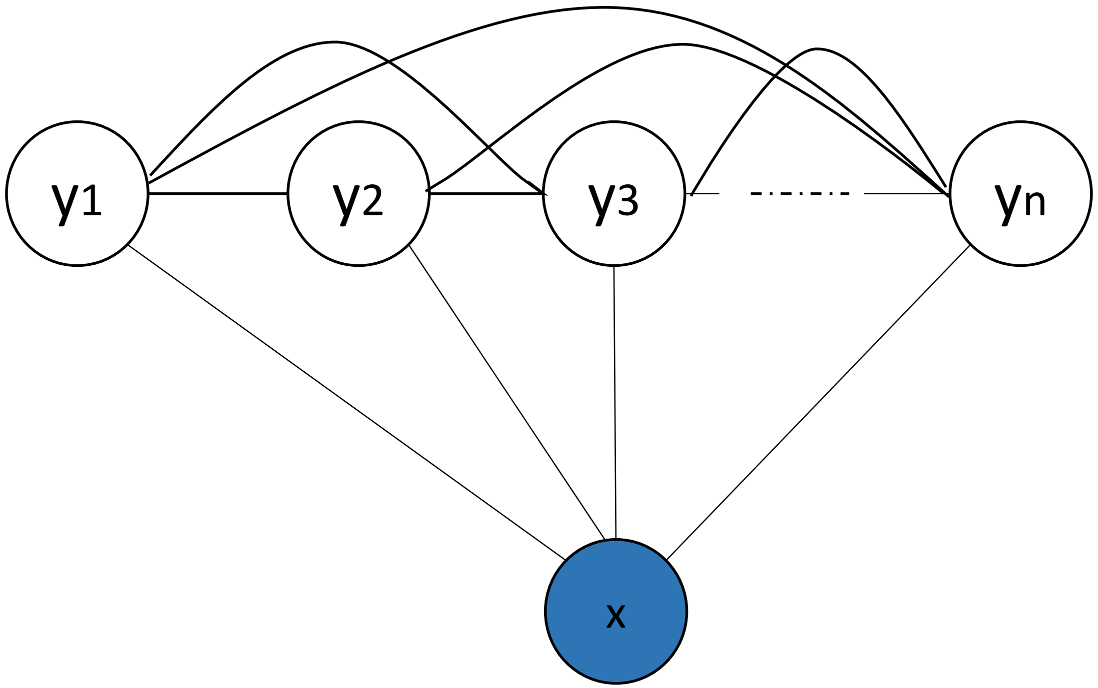
- Each output sequence has a score, which is not normalized over a particular decision

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

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

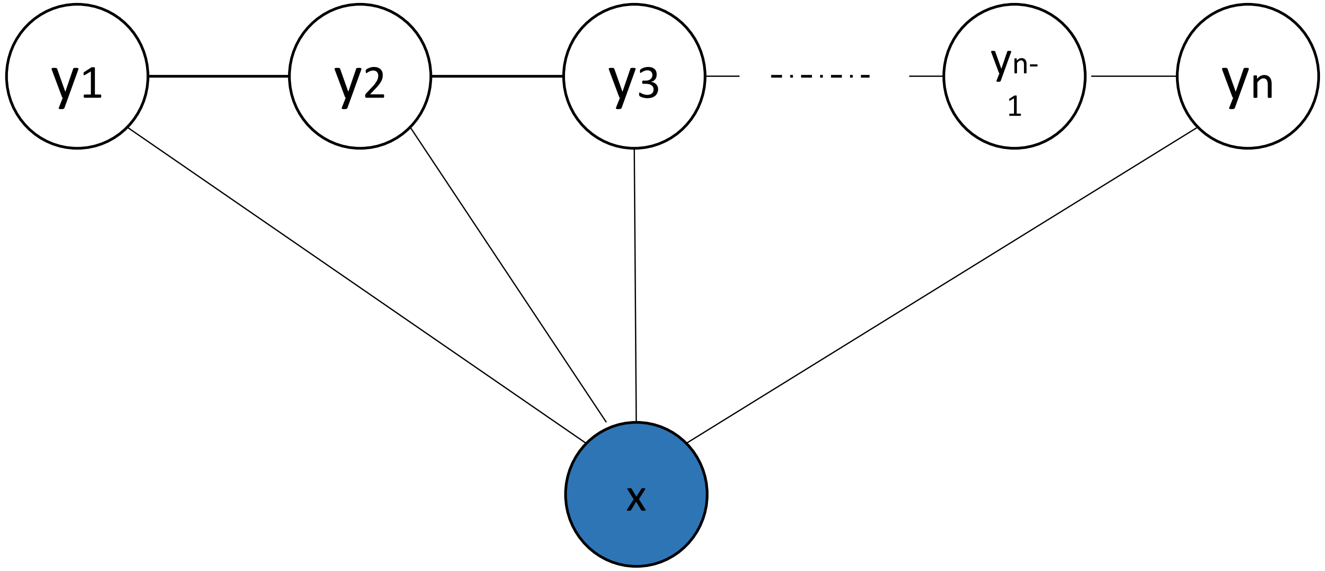
Conditional Random Fields

General form of globally normalized model



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

First-order linear CRF

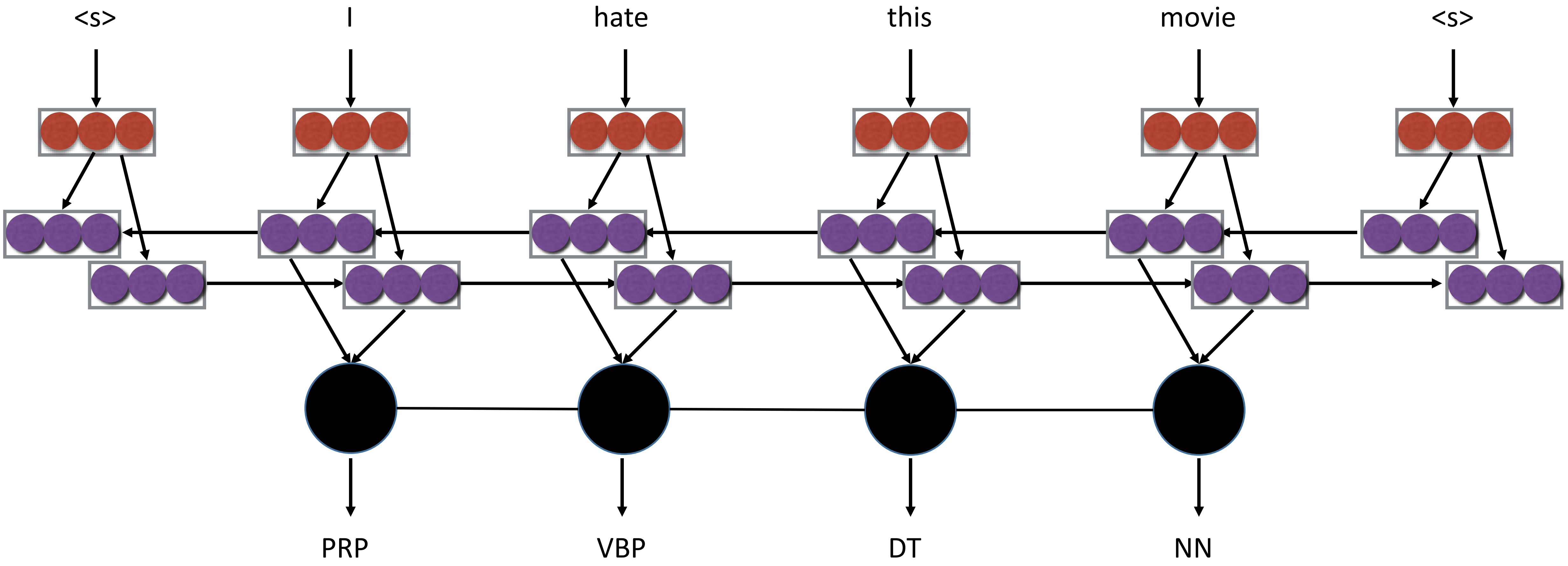


$$P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)}$$

Potential Functions

- $\psi_i(y_{i-1}, y_i, X) = \exp\left(W^T T(y_{i-1}, y_i, X, i) + U^T\ S(y_i, X, i) + b_{y_{i-1}, y_i}\right)$
  - Using neural features in DNN:
    $$\psi_i(y_{i-1}, y_i, X) = \exp\left(W^T_{y_{i-1}, y_i} F(X, i) + U^T_{y_i}\ F(X, i) + b_{y_{i-1}, y_i}\right)$$
    - Number of parameters: $O(|Y|^2 d_F)$
  - Simpler version:
    $$\psi_i(y_{i-1}, y_i, X) = \exp\left(W_{y_{i-1}, y_i} + U^T_{y_i} F(X, i) + b_{y_{i-1}, y_i}\right)$$
    - Number of parameters: $O(|Y|^2 + |Y| d_F)$

BiLSTM-CRF for Sequence Labeling



Training &Decoding of CRF

Viterbi Algorithm

CRF Training & Decoding

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

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

- Decoding

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

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

Interactions

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

- Each label depends on the input, and the nearby labels
- But given *adjacent* labels, others do not matter
- If we knew the score of every sequence $y_1, \ldots, y_{n-1}$, we could compute easily the score of sequence $y_1, \ldots, y_{n-1}, y_n$
- So we really only need to know the score of all the sequences ending in each y_{n-1}
- Think of that as some “precalculation” that happens before we think about y_n

Viterbi Algorithm

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

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

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

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

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

Step: Initial Part

- 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)

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

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

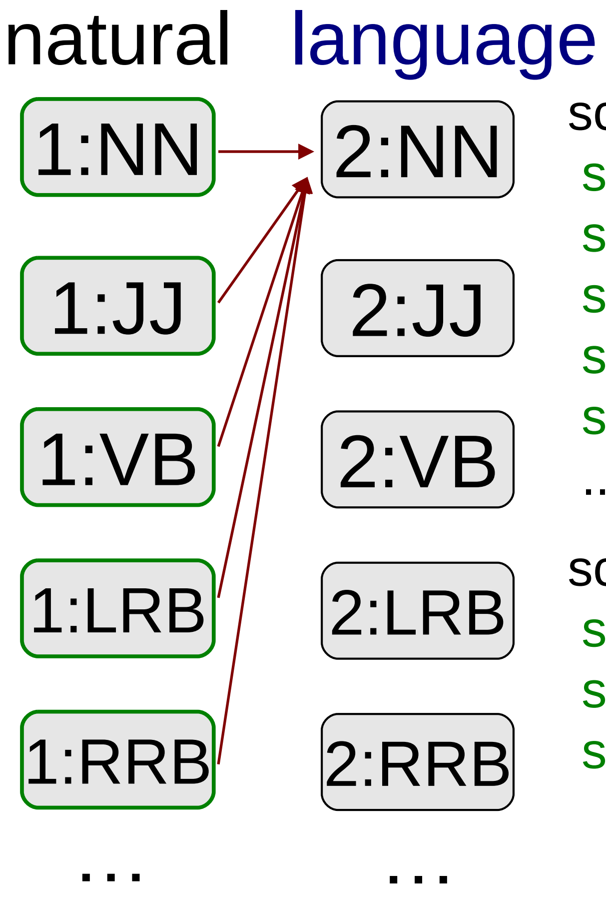
1:LRB score[“1 LRB”] = T(LRB|<S>) + S(natural | LRB)

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

…

Step: Middle Parts

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



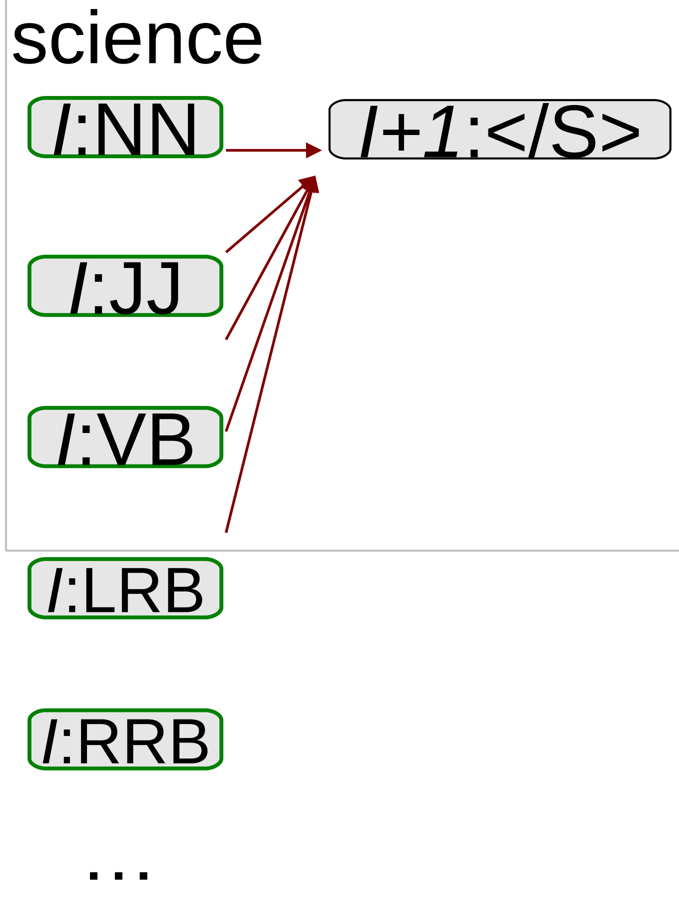
score["2 NN"] = log_sum_exp(
score["1 NN"] + T(NN|NN) + S(language | NN),
score["1 JJ"] + T(NN|JJ) + S(language | NN),
score["1 VB"] + T(NN|VB) + S(language | NN),
score["1 LRB"] + T(NN|LRB) + S(language | NN),
score["1 RRB"] + T(NN|RRB) + S(language | NN),
...)

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



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),
...
)

Viterbi Algorithm

- 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

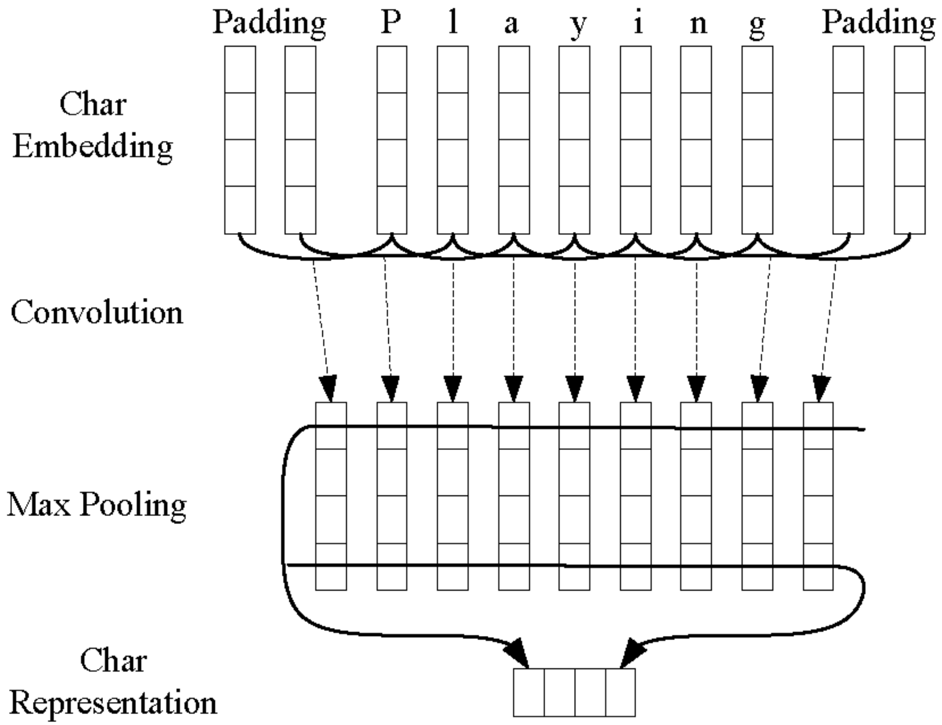
BiLSTM-CNN-CRF for Sequence Labeling

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

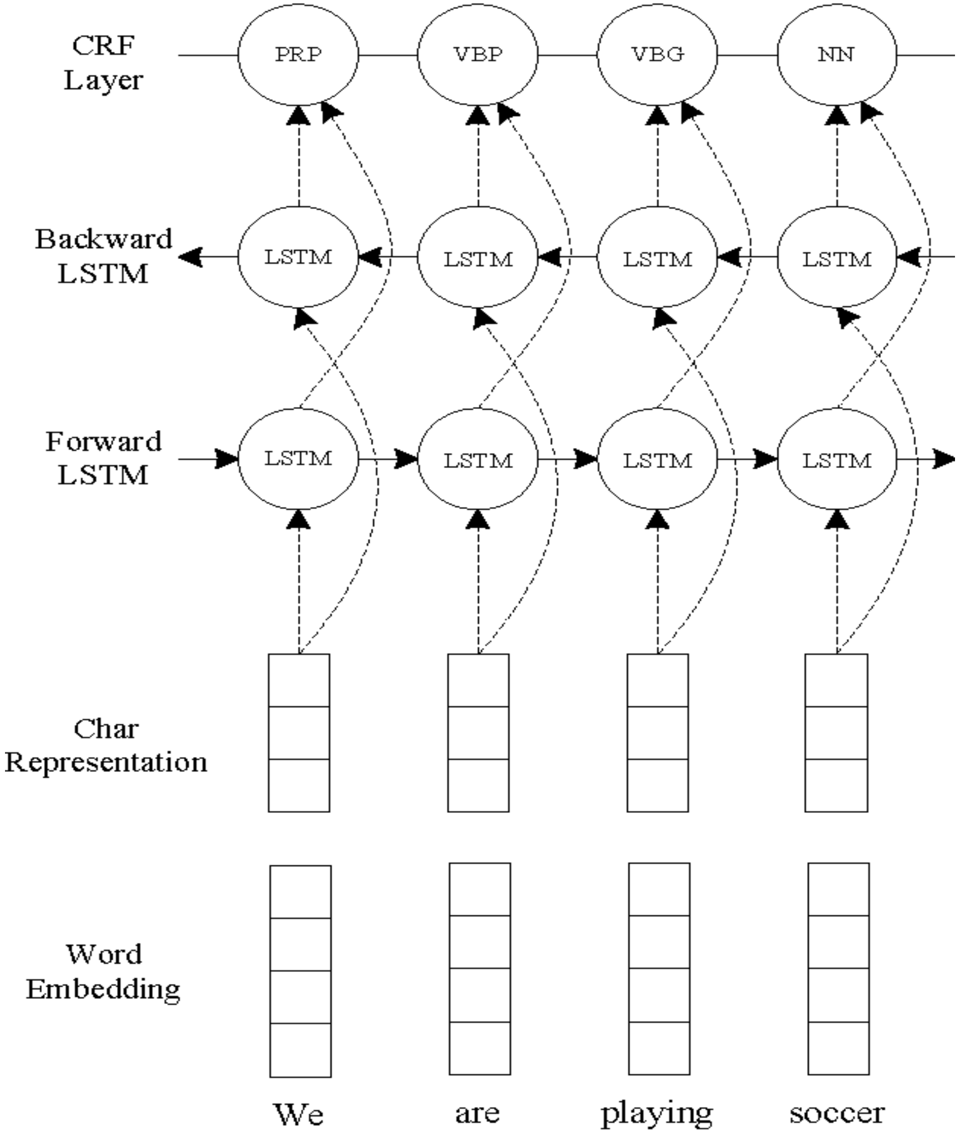
CNN for Character-level representation

- 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

Considering Rewards during Training

Reward Functions in Structured Prediction

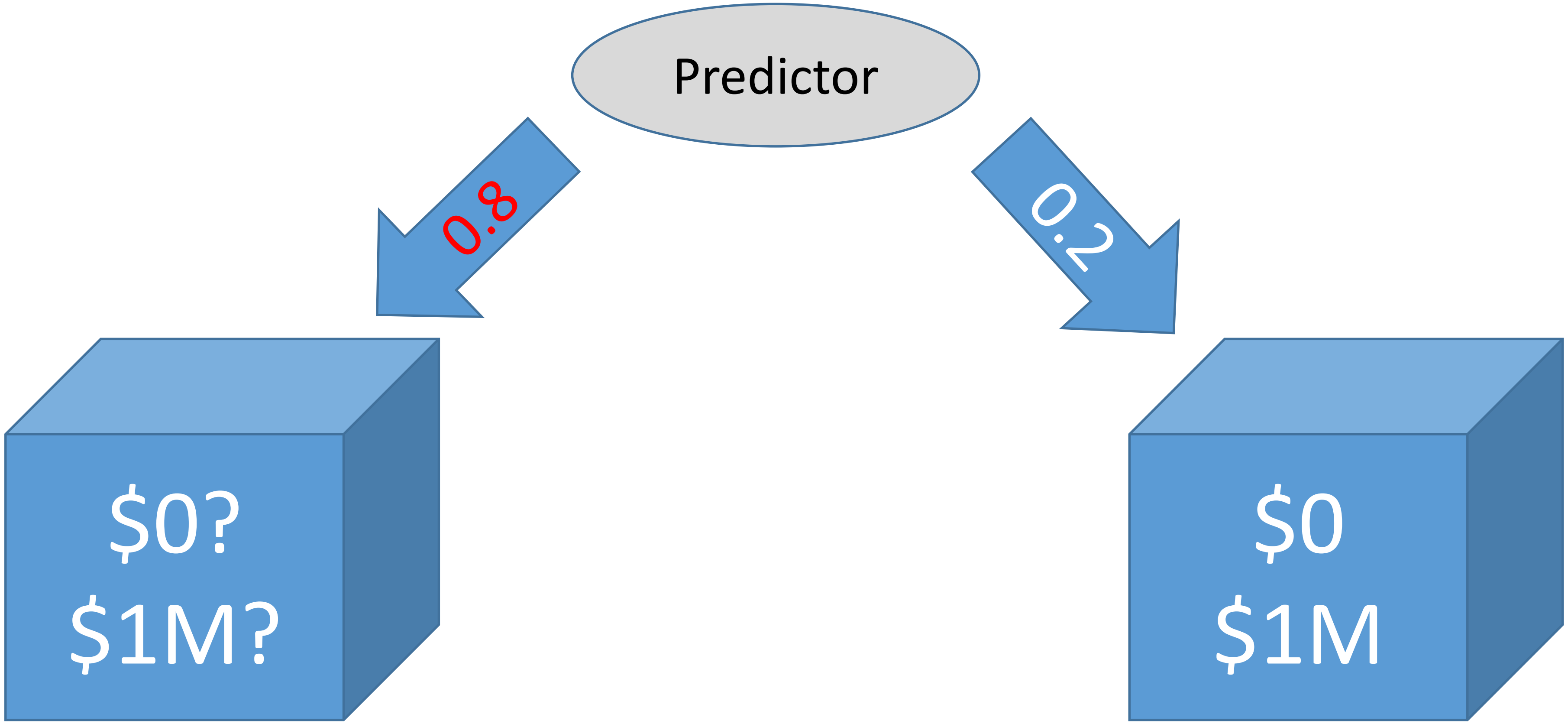
- POS tagging: token-level accuracy
- NER: F1 score
- Dependency parsing: labeled attachment score
- Machine translation: corpus-level BLEU

Do different reward functions impact our decisions?

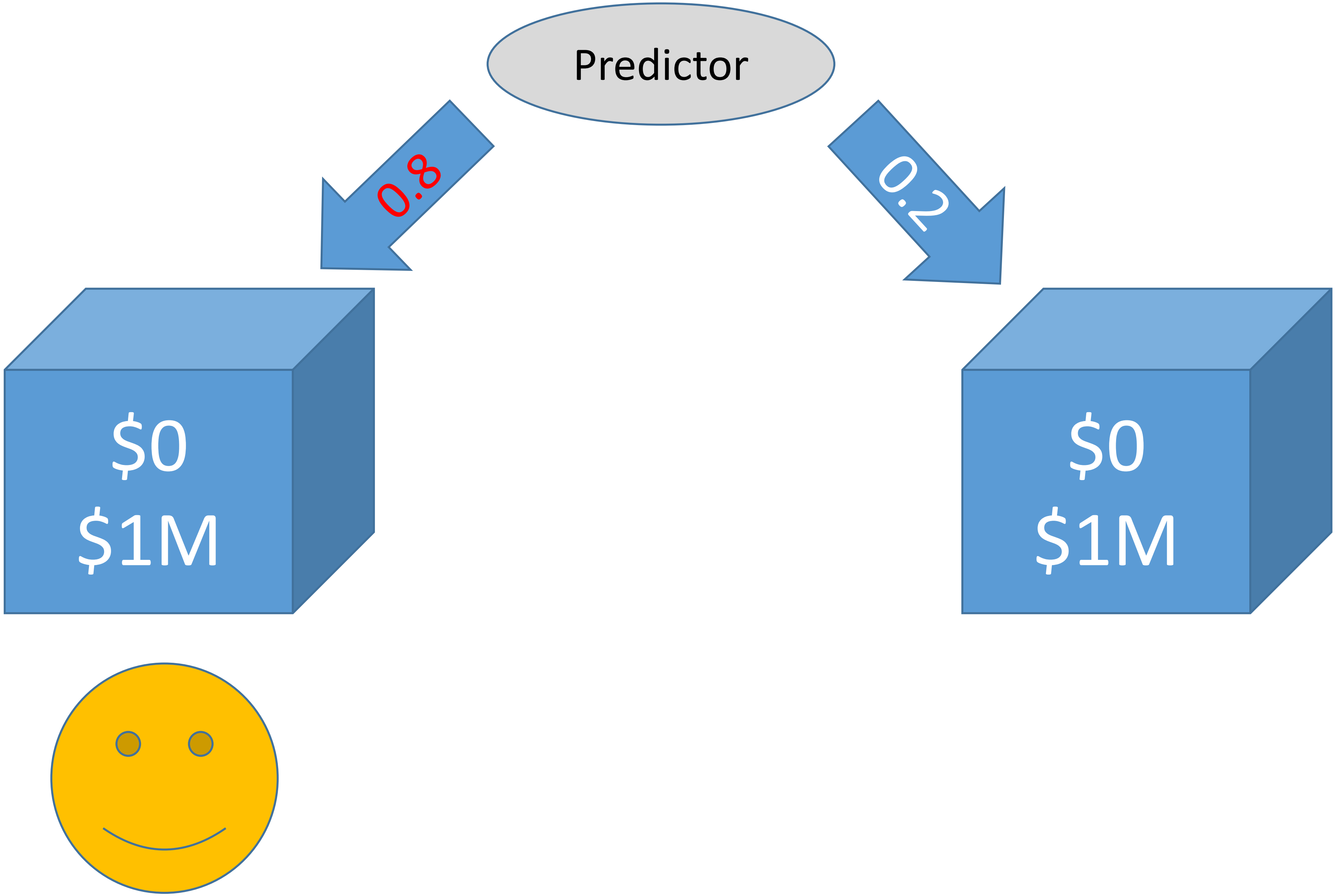
- Data1: $(X, Y) \sim P$
- Task1: predict Y given X i.e. $h_1(X)$
- Reward1: $R_1(h_1(X), Y)$

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

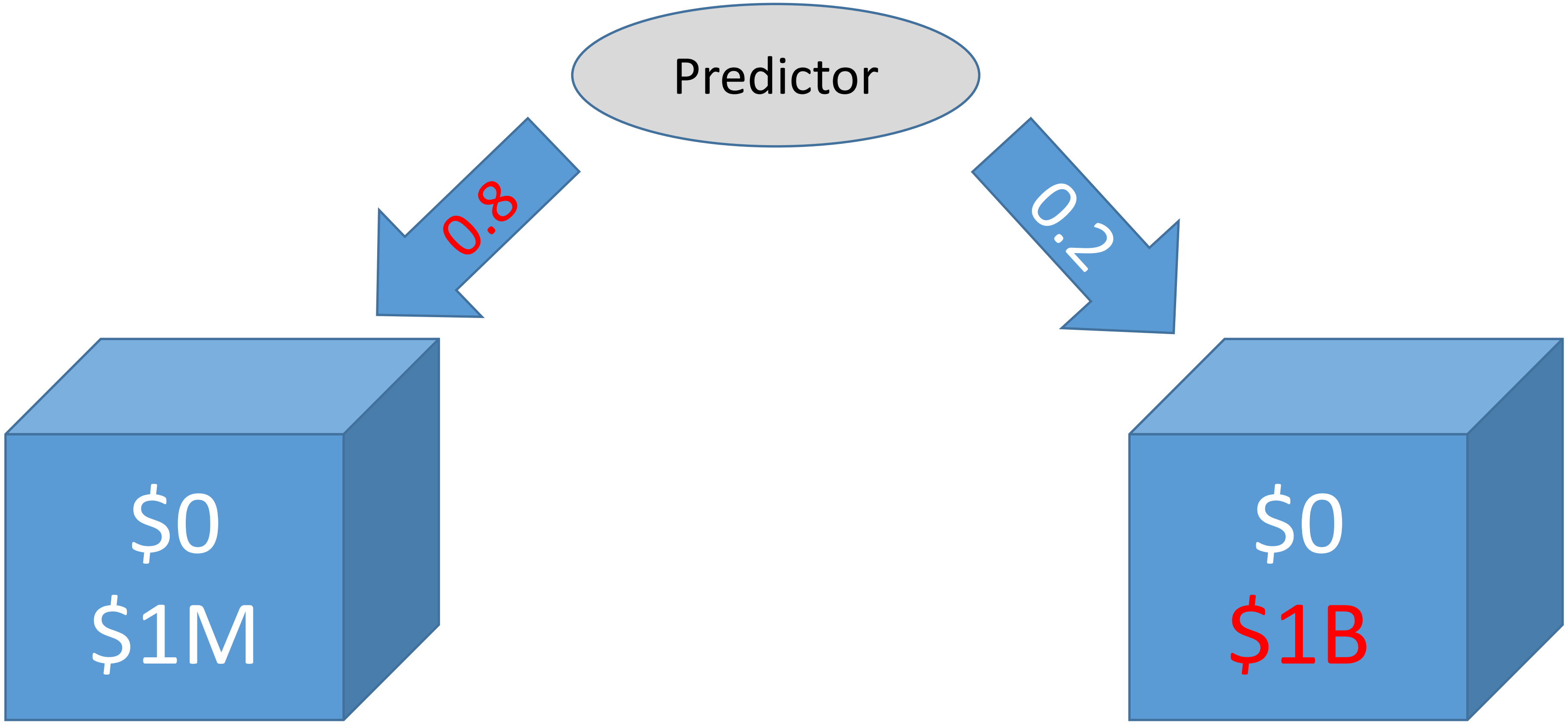
$$h_1(X) = h_2(X)?$$



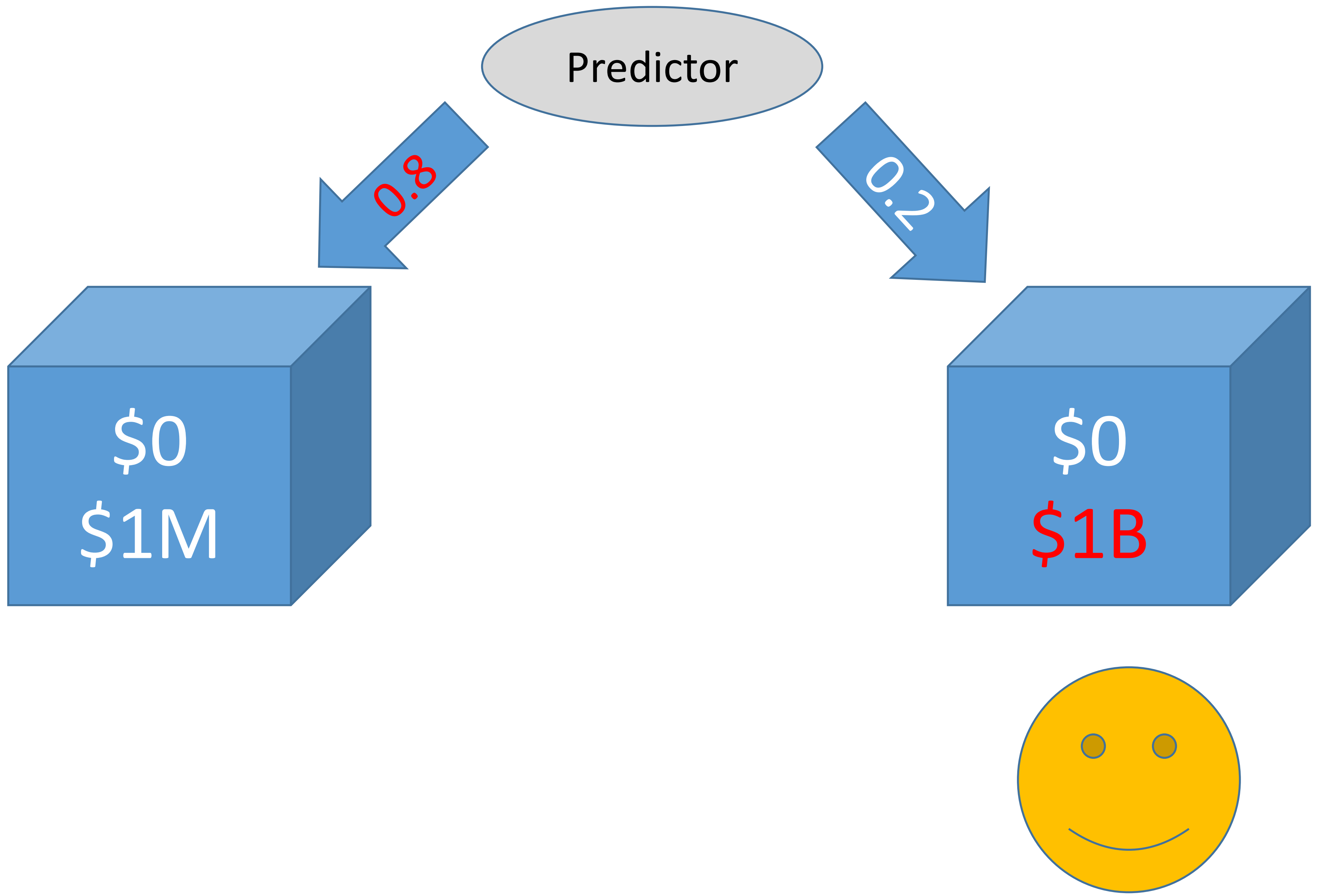
Reward is the amount of money we get



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

Considering Rewards during Training

- Max-Margin (Taskar et al., 2004)
    - Similar to cost-augmented hinge loss (last class)
    - Do not rely on a probabilistic model (only decoding algorithm is required)
- Minimum Risk Training (Shen et al., 2016)
- Reward-augmented Maximum Likelihood (Norouzi et al., 2016)

Minimum Risk Training

$$l_{MRT}(x, y; \theta) = E_{P(Y|X=x;\, \theta)}[-R(Y, y)]$$

- Pros:
  - Direct optimization w.r.t. evaluation metrics
  - Similar to the globally normalized model in (Andor et al, 2016), but with task-specific reward R
  - Applicable to arbitrary risk functions: R is not necessarily differentiable
- Cons:
  - Intractable computation of expectation w.r.t. $P(Y|X; \theta)$
  - Sampling from a sub-space

Reward-augmented Maximum Likelihood

- Reward-augmented Maximum Likelihood (RAML)
- Basic idea: randomly sample incorrect training data from the *exponentiated payoff distribution q*, train w/ maximum likelihood

$$q(y|y^*;\tau) = \frac{\exp(R(y, y^*/\tau)}{\sum_{y'} \exp(R(y', y^*/\tau)}$$

I hate this movie

MLE

PRP **NN** DT NN

sample

PRP VBP DT NN

Can be shown to approximately maximize reward, Norouzi et al., (2016) and Ma et al. (2017)

Questions?