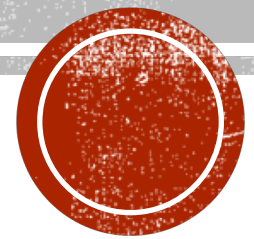
STRUCTURE LEARNING IN NLP

Presented By Yan Li



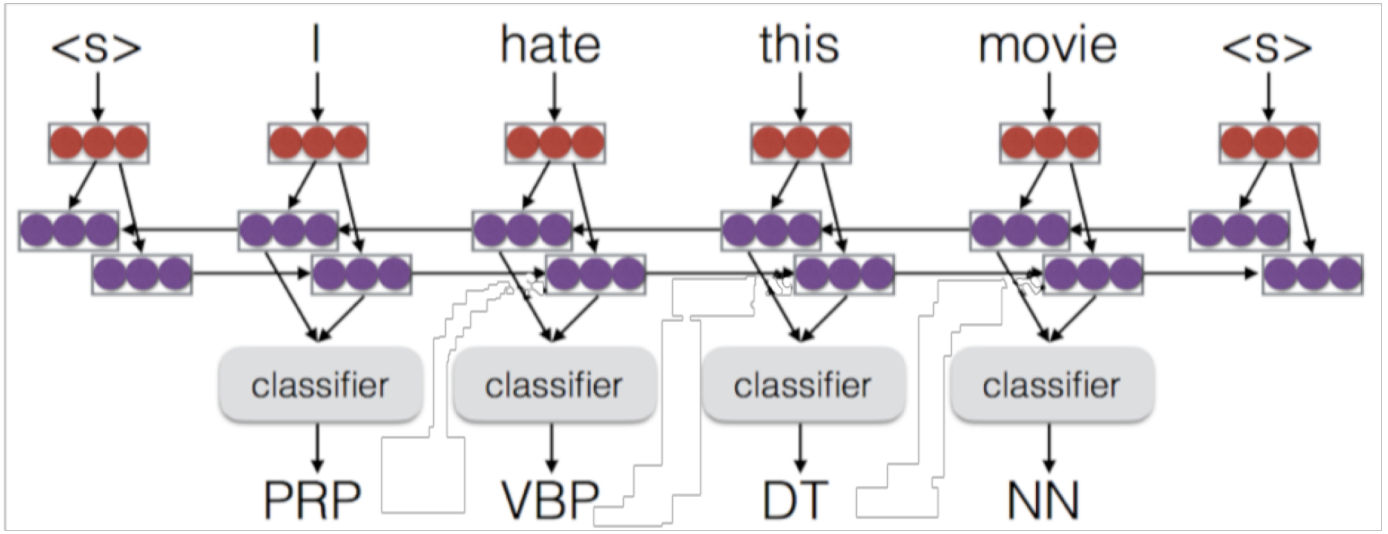
MOTIVATION

- In NLP or other tasks (i.e., image segmentation) the output is not just a value but have certain structure, and encoding the relationship among outputs can improve performance a lot.

- For example, in POS (part of speech) tagging usually a verb cannot followed by another verb. If this kind of information can be encoded in the model, then the performance will be improved.

CASE STUDY (POS TAGGING)

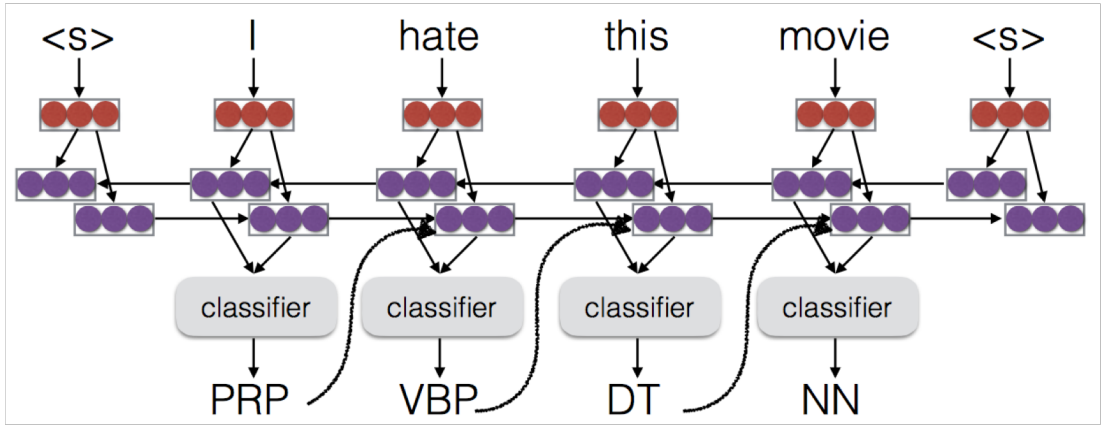
- POS tagging : Determine the characteristic or property of a certain word in a sentence, i.e., whether a certain word is verb, noun, adjective, auxiliary word, …

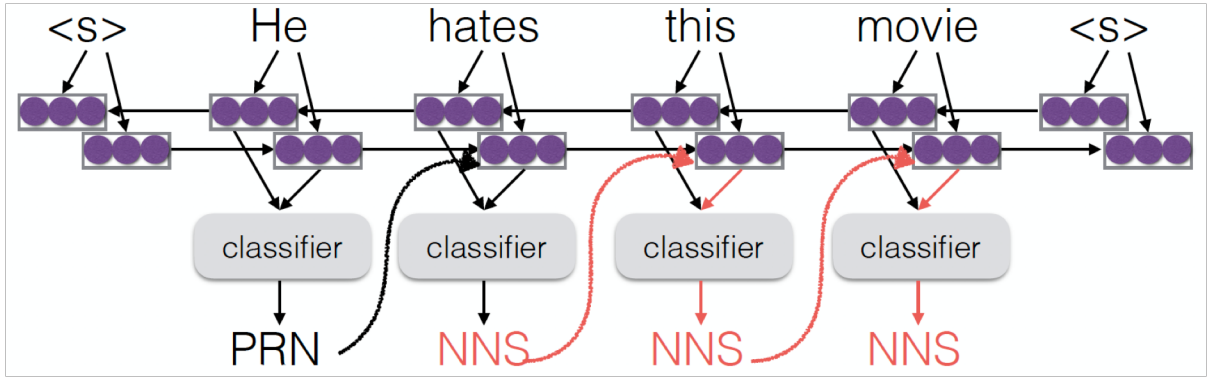


- Structured prediction task, but not structured prediction model: multi-class classification

POS TAGGING WITH TEACHING FORCING

- 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 MODEL V.S. GLOBAL MODEL

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

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

Time complexity |Y|*k

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

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

Time complexity $k^{|Y|}$

STRUCTURED PERCEPTRON

- Idea: the score of True label-data combination is grater than any other combination:
- How to achieve:
  1. Find the best prediction under current θ

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

  2. Then update θ according to

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

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

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

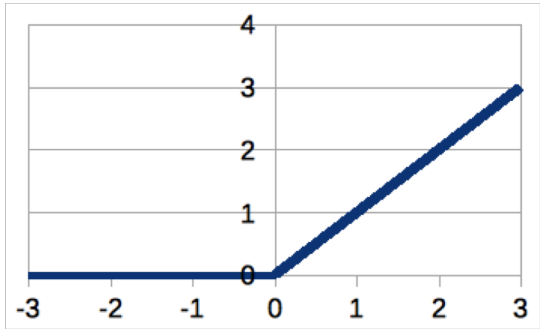
- This is a normal loss function, can be used in NNs
- But! Requires finding the argmax in addition to the true candidate: must do prediction during training

STRUCTURED SVM

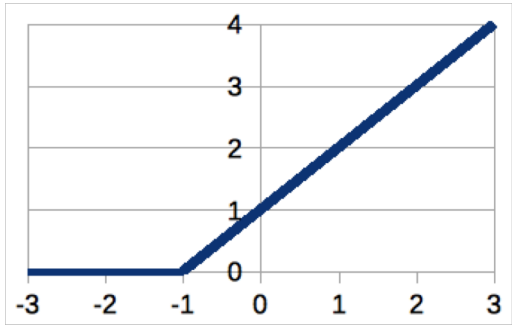
- Add margin and error in Structure perceptron to make it more robust, i.e., the score of True label-data combination is grater than any other combination and the True label is far from the others.
- Error function $cost(\tilde{Y} - Y)$ tells the difference between $\tilde{Y}$ and Y.
- How to achieve:

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

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



Perceptron



Hinge

COSTS OVER SEQUENCES

- Zero-one loss: 1 if sentences differ, zero otherwise

$$\text{cost}_{\text{zero-one}}(\hat{Y}, Y) = \delta(\hat{Y} \neq Y)$$

- Hamming loss: 1 for every different element (lengths are identical)

$$\text{cost}_{\text{hamming}}(\hat{Y}, Y) = \sum_{j=1}^{|Y|} \delta(\hat{y}_j \neq y_j)$$

- Other losses: edit distance, 1-BLEU, etc.

STRUCTURED TRAINING AND PRE-TRAINING

- Neural network models have lots of parameters and a big output space; training is hard
- Tradeoffs between training algorithms:
  - Selecting just one negative example is inefficient
  - Teacher forcing efficiently updates all parameters, but suffers from exposure bias, label bias
- Thus, it is common to pre-train with teacher forcing, then fine-tune with more complicated algorithm

CONTRASTING PERCEPTRON AND GLOBAL NORMALIZATION

- Globally normalized probabilistic model

$$\ell_{\text{global}}(X, Y; \theta) = -\log \frac{e^{S(Y|X)}}{\sum_{\tilde{Y}} e^{S(\tilde{Y}|X)}}$$

- Structured perceptron

Boost up the score of True label-data pair just based on the current highest score.

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

- Global structured perceptron?

Boost up the score of True label-data pair based on the current score of all label-data pairs.

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

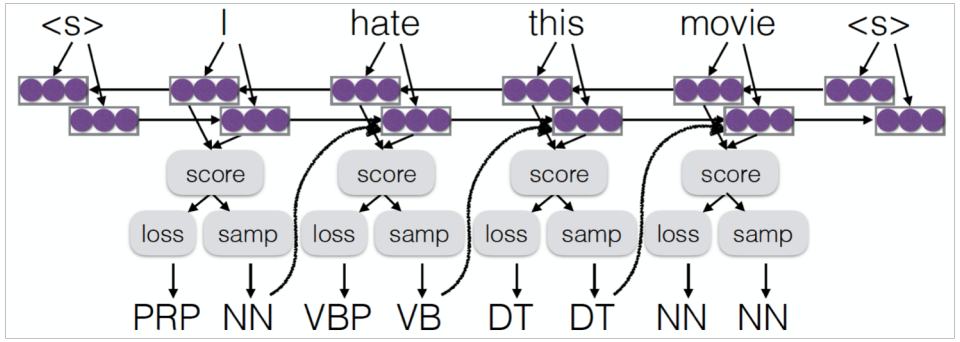
  - Same computational problems as globally normalized probabilistic models

DRAWBACK OF ABOVE TWO METHODS

- Considers fewer hypotheses, so unstable
- Requires decoding, so slow
- Generally must resort to pre-training (and even then, it's not as stable as teacher forcing w/ MLE)

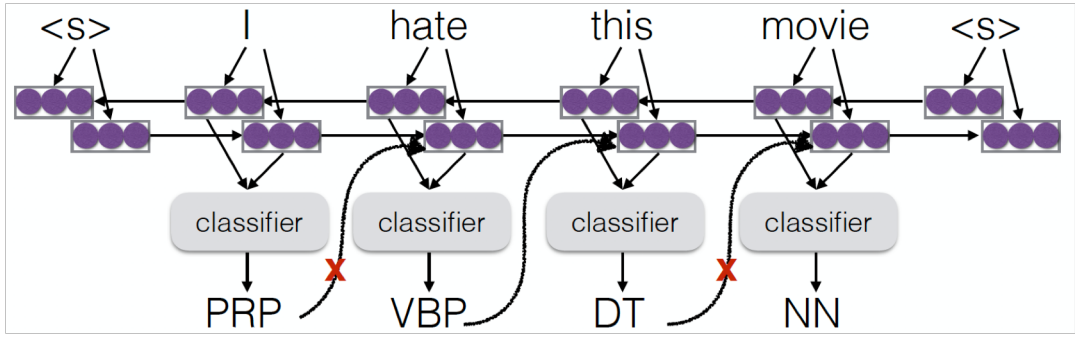
POSSIBLE ALTERNATIVE SOLUTIONS

- Solution 1: Sample Mistakes in Training
  - DAgger, also known as "scheduled sampling", etc., randomly samples wrong decisions and feeds them in



Start with no mistakes, and then gradually introduce them using annealing

- Solution 2: Drop Out Inputs
  - Simply don't input the previous decision sometimes during training (Gal and Ghahramani 2015)



Helps ensure that the model doesn't rely too heavily on predictions, while still using them

CONDITIONAL RANDOM FIELDS (CRF)

- 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.

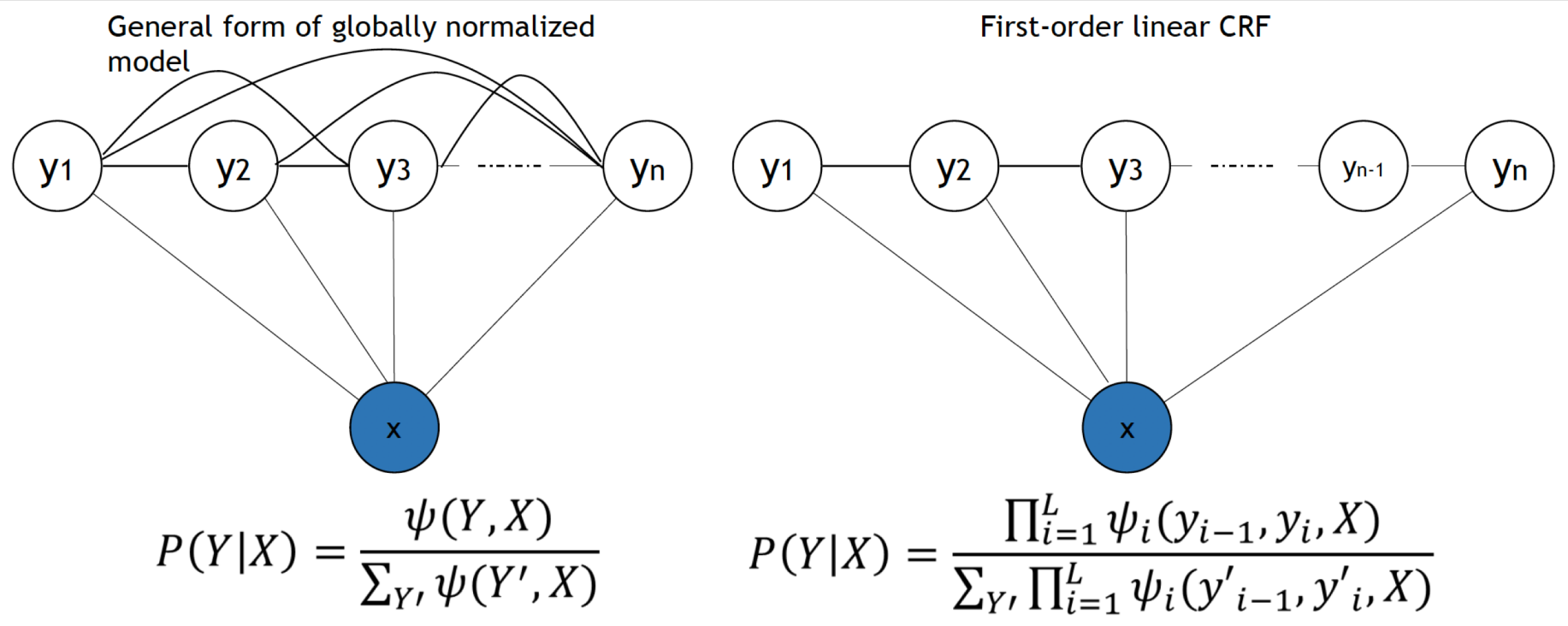
Training:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} -\log[P(Y|X)]$$

An extension of cross entropy

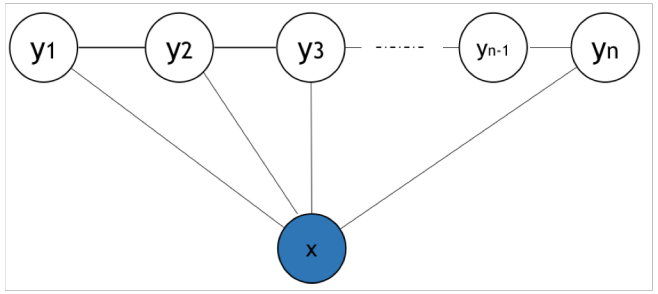
CRF is an extension of logistic regression, in logistic regression each output is a vector, while in CRF each output is a structure with multiple components.

CRF IN PRACTICE



- Add independent assumption In graphical model to simplify the problem. In the first order linear CRF the Viterbi Algorithm can be used in calculation

FIRST ORDER LINEAR CRF



- Potential Functions

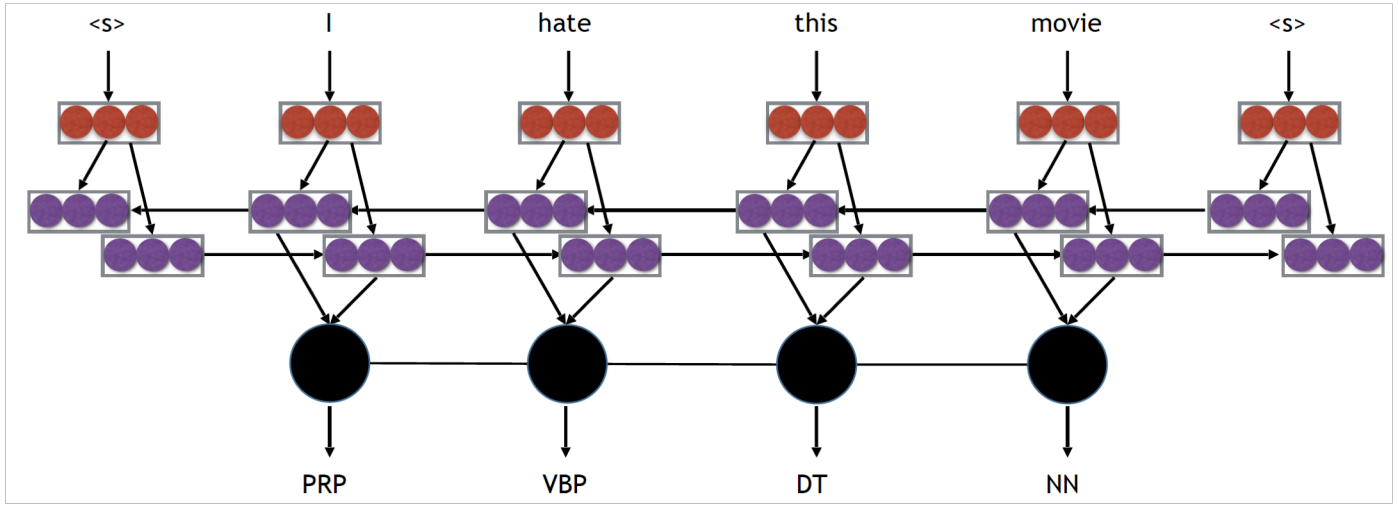
"Transition" "Emission"

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

From last out put to current output

The score of current output

- BiLSTM-CRF for Sequence Labeling



COMPUTING THE PARTITION FUNCTION IN FIRST ORDER LINEAR CRF

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

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

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

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

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

The current π can be recursively commutated; therefore, the dynamic programming algorithm (Viterbi Algorithm) can be used to efficiently update π

DECODING AND GRADIENT CALCULATION IN FIRST ORDER LINEAR CRF

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

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

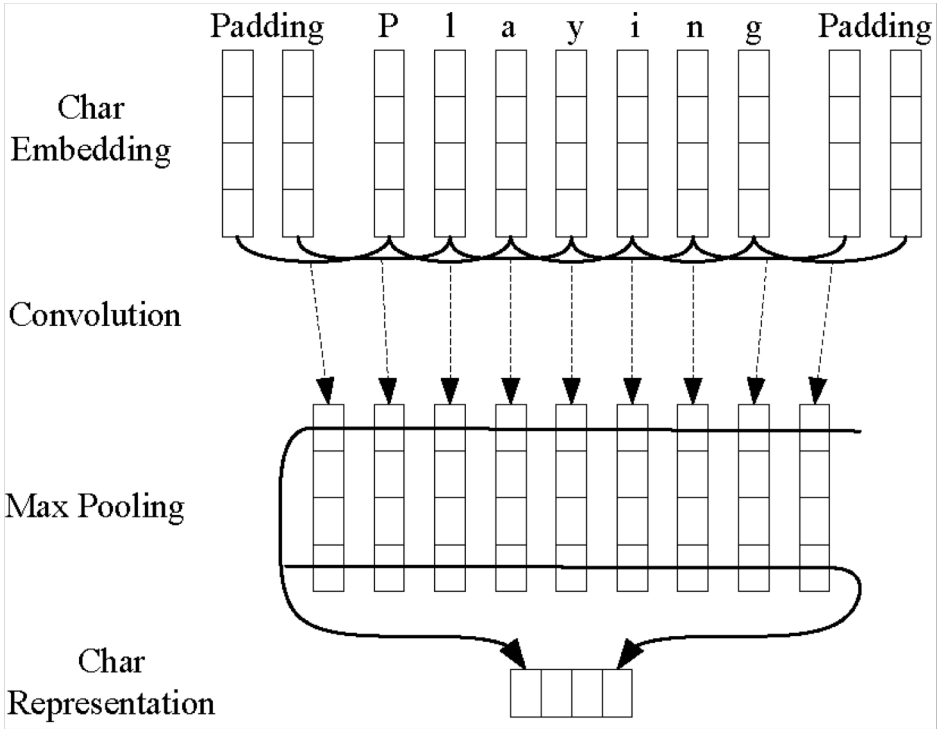
  - Forward-backward algorithm (Sutton and McCallum, 2010)
    - Both $P(Y|X; \theta)$ and $F(Y, X)$ can be decomposed
    - Need to compute the marginal distribution:

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

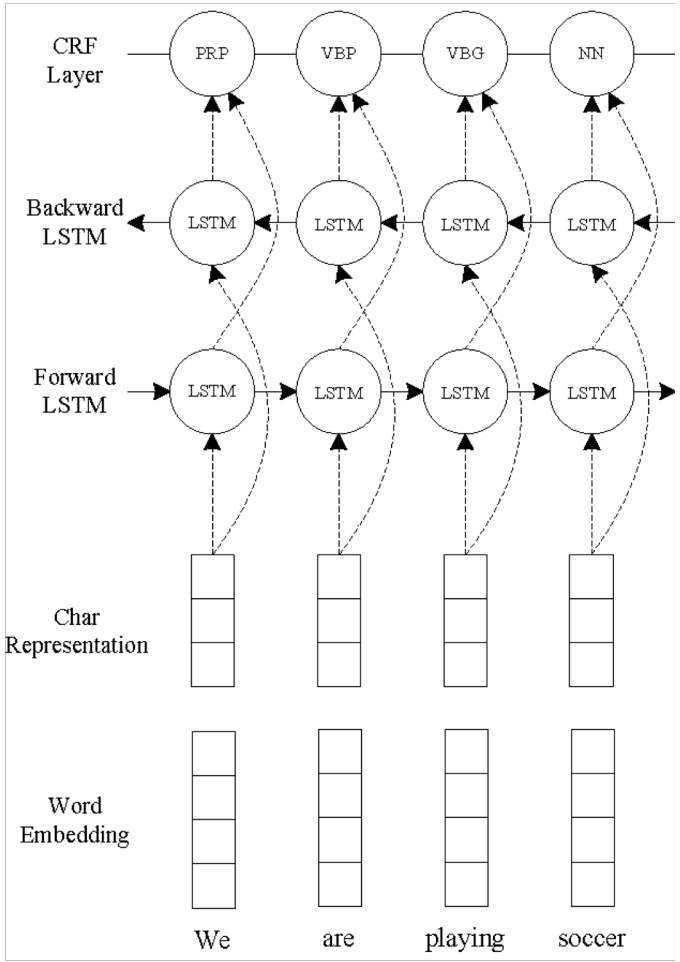
  - Not necessary if using DNN framework (auto-grad)

BILSTM-CNN-CRF FOR SEQUENCE LABELING

- CNN for Character-level representation



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



COMPARE CRF AND STRUCTURED PERCEPTRON

- ***Structured Perceptron***

$$\tilde{y}^n = \arg\max_y w \cdot \phi(x^n, y)$$

$$w \rightarrow w + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$

Hard

- ***CRF***

$$w \rightarrow w + \eta \left(\phi(x^n, \hat{y}^n) - \sum_{y'} P(y'|x^n)\, \phi(x^n, y') \right)$$

Soft

UNIFIED FRAMEWORK IN STRUCTURE LEARNING

Training

- Find a function F

$$F : X \times Y \rightarrow R$$

- F(x,y): evaluate how compatible the objects x and y is

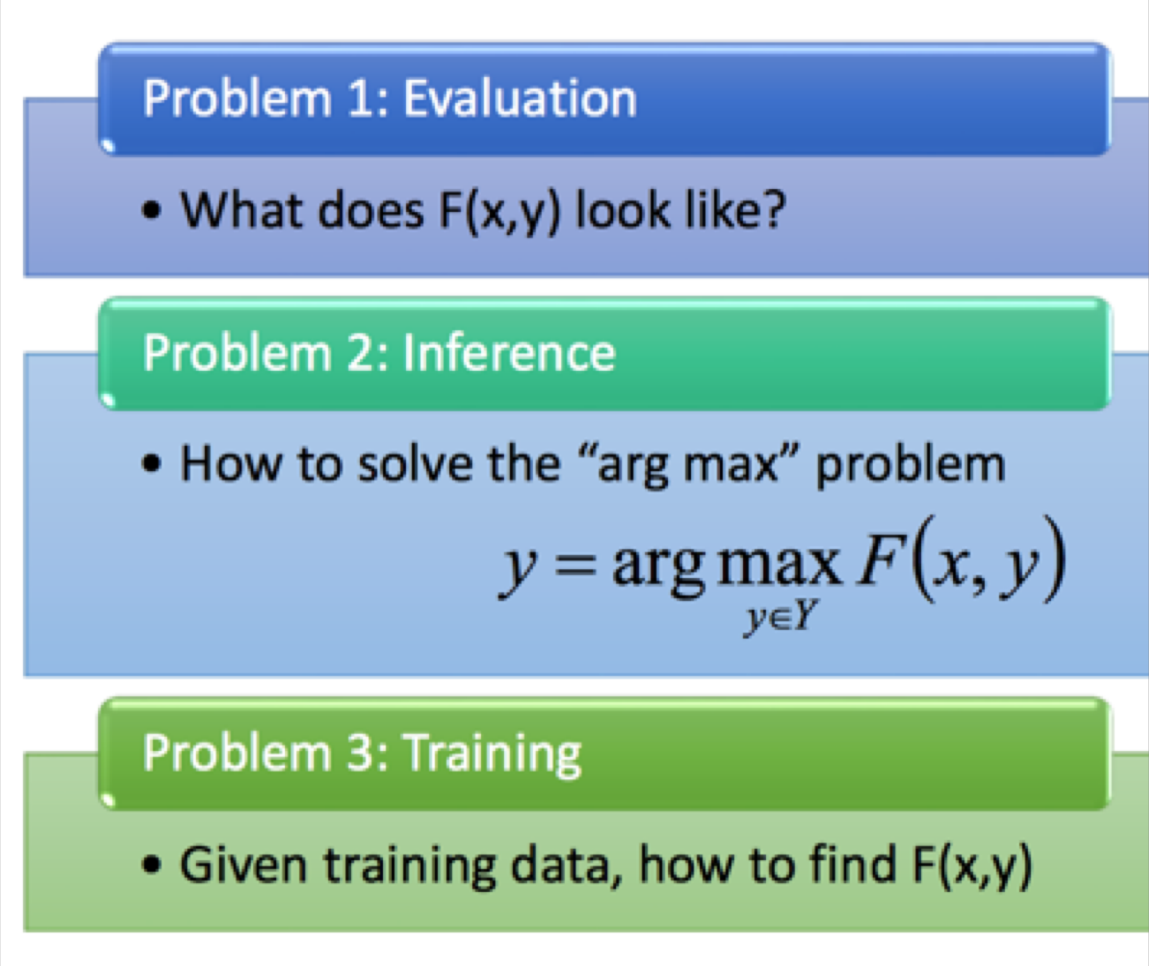
Inference (Testing)

- Given an object x

$$\tilde{y} = \arg\max_{y \in Y} F(x, y)$$

In some problem F(x, y) can be viewed as joint probability, Note here F(x, y) is S(x, y) in previous slides

THREE KEY PROBLEMS IN STRUCTURE LEARNING



- In general, structure learning is a hard problem comparing with the traditional non-structural learning problem.

REFERENCE

- Structured Perceptron (Collins 2002)
- Structured Hinge Loss (Taskar et al. 2005)
- SEARN (Daume et al. 2006)
- DAgger (Ross et al. 2011)
- Dynamic Oracles (Goldberg and Nivre 2013)
- Training Neural Parsers w/ Dynamic Oracles (Ballesteros et al. 2016)
- Word Dropout (Gal and Ghahramani 2015)
- RAML (Norouzi et al. 2016)
- An Introduction to CRFs (Sutton and McCallum 2011)
- Bidirectional LSTM-CRF Models for Sequence Tagging (Huang et al. 2015)
- Conditional Random Fields (Lafferty et al. 2001)
- Minimum Risk Training for Neural MT (Shen et al. 2016)
- Globally Normalized Networks (Andor et al. 2016)
- Reward Augmented Maximum Likelihood (Norouzi et al. 2016)
- Softmax Q-Distribution Estimation (Ma et al. 2016)
- End-to-end Sequence Labeling with BiLSTM-CNN-CRF (Ma et al. 2016)

THANKS

Q&A