COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

# Sequence Labeling: more tasks, more methods

**CMSC 470**

Marine Carpuat

# Recap: We know how to perform POS tagging with structured perceptron

- An example of sequence labeling tasks
- Requires a predefined set of POS tags
  - Penn Treebank commonly used for English
  - Encodes some distinctions and not others
- Given annotated examples, we can address sequence labeling with multiclass perceptron
  - but computing the argmax naively is expensive
  - constraints on the feature definition make efficient algorithms possible
  - Viterbi algorithm for unary and markov features

# Sequence labeling tasks

Beyond POS tagging

# Many NLP tasks can be framed as sequence labeling

- Information Extraction: detecting named entities
  - E.g., names of people, organizations, locations

"Brendan Iribe, a co-founder of Oculus VR and a prominent University of Maryland donor, is leaving Facebook four years after it purchased his company."

http://www.dbknews.com/2018/10/24/brendan-iribe-facebook-leaves-oculus-vr-umd-computer-science/

# Many NLP tasks can be framed as sequence labeling

x = [Brendan, Iribe, ",",  a, co-founder, of, Oculus, VR, and, a, prominent, University, of, Maryland, donor, ",", is, leaving, Facebook, four, years, after, it, purchased, his, company, "."]

y = [B-PER, I-PER, O, O, O, O, B-ORG, I-ORG, O, O, O,B-ORG, I-ORG, I-ORG, O, O, O,B-ORG, O, O, O, O, O, O, O, O]

**"BIO" labeling scheme for named entity recognition**

# Many NLP tasks can be framed as sequence labeling

- The same kind of BIO scheme can be used to tag other spans of text
  - Syntactic analysis: detecting noun phrase and verb phrases
  - Semantic roles: detecting semantic roles (who did what to whom)

# Many NLP tasks can be framed as sequence labeling

- Other sequence labeling tasks
  - Language identification in code-switched text
    "Ulikuwa ukiongea a lot of nonsense." (Swahili/English)
  - Metaphor detection
    "he **swam** in a **sea** of diamonds"
    "authority is a **chair**, it needs **legs** to **stand**"
    "in Washington, people change **dance partners** frequently, but not the **dance"**
  - …

# Other algorithms for solving the argmax problem

# Structured perceptron can be used for other structures than sequences

- The Viterbi algorithm we’ve seen is specific to sequences
  - Other argmax algorithms necessary for other structures (e.g. trees)
- Integer Linear Programming provides a general framework for solving the argmax problem

# Argmax problem as an Integer Linear Program

- An integer linear program (ILP) is an optimization problem of the form

$$\max_{\boldsymbol{z}} \quad \boldsymbol{a} \cdot \boldsymbol{z} \quad \text{subj. to} \quad \text{linear constraints on } \boldsymbol{z}$$

  - For a fixed vector a
  - Example of integer constraint: $z_3 \in \{0, 1\}$

- Well-engineered solvers exist
  - e.g, Gurobi
  - Useful for prototyping
  - But general not as efficient as dynamic programming

# Casting sequence labeling with Markov features as an ILP

- Step 1: Define variables z as binary indicator variables which encode an output sequence y

$$z_{l,k',k} = \mathbf{1}[\text{label } l \text{ is } k \text{ and label } l-1 \text{ is } k']$$

- Step 2: Construct the linear objective function

$$a_{l,k',k} = \boldsymbol{w} \cdot \phi_l(\boldsymbol{x}, \langle \ldots, k', k \rangle)$$

# Casting sequence labeling with Markov features as an ILP

- Step 3: Define constraints to ensure a well-formed solution
  - Z's should be binary: for all l, k', k

$$z_{l,k',k} \in \{0,1\}$$

  - For a given position l, there is exactly one active z

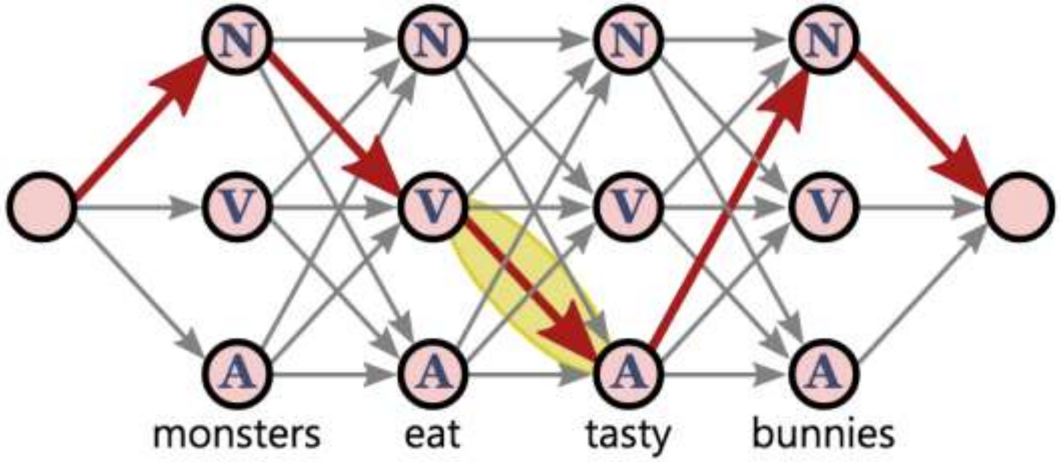
$$\sum_k \sum_{k'} z_{l,k',k} = 1 \text{ for all } l$$

  - The z's are internally consistent

$$\sum_{k'} z_{l,k',k} = \sum_{k''} z_{l+1,k,k''} \text{ for all } l, k$$

# Loss-augmented structured prediction

# In default structured perceptron, all bad output sequences are equally bad




- Consider
- $\widehat{y_1} = [A, A, A, A]$
- $\widehat{y_2} = [N, V, N, N]$

- With 0-1 loss

  $l^{(0-1)}(y, \widehat{y_1}) = l^{(0-1)}(y, \widehat{y_2}) = 1$

- An alternative
  - **Hamming Loss** gives a more nuanced evaluation of output than 0–1 loss

$$\ell^{(\text{Ham})}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{l=1}^{L} \mathbf{1}[y_l \neq \hat{y}_l]$$

# Loss functions for structured prediction

- Recall learning as optimization for multiclass classification
  - e.g., $\min_{w} \quad \frac{1}{2}||w||^2 + C\sum_n \ell^{(\text{hin})}(y_n, w \cdot x_n + b)$
- Let's define a structure-aware optimization objective
  - e.g., $\min_{w} \quad \frac{1}{2}||w||^2 + C\sum_n \ell^{(\text{s-h})}(y_n, x_n, w)$

$$\ell^{(\text{s-h})}(y_n, x_n, w) = \max\left\{0, \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[s_w(x_n, \hat{y}) + \ell^{(\text{Ham})}(y_n, \hat{y})\right] - s_w(x_n, y_n)\right\}$$

**Structured hinge loss**

- 0 if true output beats score of every imposter output
- Otherwise: scales linearly as function of score diff between most confusing imposter and true output

# Optimization: stochastic *sub*gradient descent

- Subgradients of structured hinge loss?

$$= \max \left\{ 0, \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[ s_w(x_n, \hat{y}) + \ell^{(\text{Ham})}(y_n, \hat{y}) \right] - s_w(x_n, y_n) \right\}$$

$$\nabla_w \ell^{(\text{s-h})}(y, x, w) \quad \textit{if} \text{ the loss is } > 0 \tag{17.25}$$

expand definition using arbitrary structured loss $\ell$

$$= \nabla_w \left\{ \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[ w \cdot \phi(x_n, \hat{y}) + \ell(y_n, \hat{y}) \right] - w \cdot \phi(x_n, y_n) \right\} \tag{17.26}$$

define $\hat{y}_n$ to be the output that attains the maximum above, rearrange

$$= \nabla_w \left\{ w \cdot \phi(x_n, \hat{y}) - w \cdot \phi(x_n, y_n) + \ell(y_n, \hat{y}) \right\} \tag{17.27}$$

take gradient

$$= \phi(x_n, \hat{y}) - \phi(x_n, y_n) \tag{17.28}$$

# Optimization: stochastic subgradient descent

- subgradients of structured hinge loss

$$\nabla_{\boldsymbol{w}} \ell^{(\text{s-h})}(\boldsymbol{y}_n, \boldsymbol{x}_n, \boldsymbol{w}) = \begin{cases} \boldsymbol{0} & \text{if } \ell^{(\text{s-h})}(\boldsymbol{y}_n, \boldsymbol{x}_n, \boldsymbol{w}) = 0 \\ \phi(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n) - \phi(\boldsymbol{x}_n, \boldsymbol{y}_n) & \text{otherwise} \end{cases}$$

$$\text{where } \hat{\boldsymbol{y}}_n = \operatorname*{argmax}_{\hat{\boldsymbol{y}}_n \in \mathcal{Y}(\boldsymbol{x}_n)} \left[ \boldsymbol{w} \cdot \phi(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n) + \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) \right] \quad (17.29)$$

# Optimization: stochastic subgradient descent
# Resulting training algorithm

**Algorithm 41** $\textsc{StochSubGradStructSVM}(\mathbf{D}, \mathit{MaxIter}, \lambda, \ell)$

$w \leftarrow \mathbf{0}$ // initialize weights
**for** $\mathit{iter} = 1 \ldots \mathit{MaxIter}$ **do**
  **for all** $(x,y) \in \mathbf{D}$ **do**
    $\hat{y} \leftarrow \operatorname{argmax}_{\hat{y} \in \mathcal{Y}(x)} w \cdot \phi(x, \hat{y}) + \ell(y, \hat{y})$ // loss-augmented prediction
    **if** $\hat{y} \neq y$ **then**
      $w \leftarrow w + \phi(x, y) - \phi(x, \hat{y})$ // update weights
    **end if**
    $w \leftarrow w - \frac{\lambda}{N} w$ // shrink weights due to regularizer
  **end for**
**end for**
**return** $w$ // return learned weights

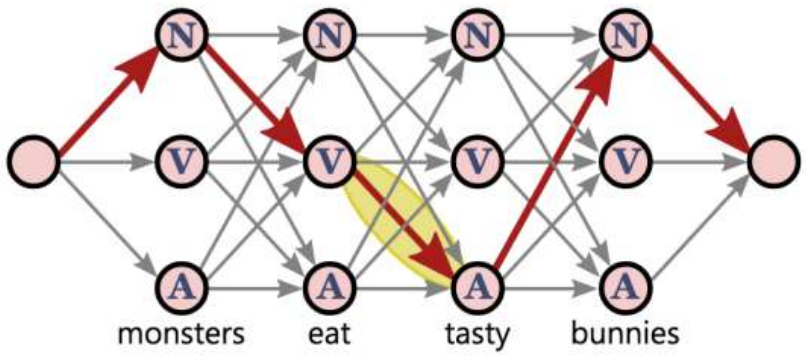
Only 2 differences compared to structured perceptron!

# Loss-augmented inference/search

## Recall dynamic programming solution without Hamming loss

$$\tilde{\alpha}_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

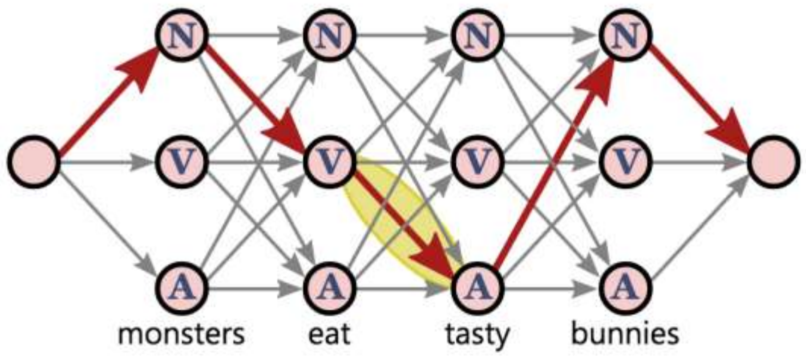
$$= \max_{k'} \left[ \tilde{\alpha}_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle) \right]$$



# Loss-augmented inference/search
# Dynamic programming with Hamming loss

$$\tilde{\alpha}_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) + \ell^{\text{(Ham)}}_{1:l+1}(\boldsymbol{y}, \hat{\boldsymbol{y}} \circ k)$$

$$= \max_{k'} \left[ \tilde{\alpha}_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \right] + \mathbf{1}[k \neq \boldsymbol{y}_{l+1}]$$




We can use Viterbi algorithm as before **as long as the loss function decomposes over the input consistently with features!**

# Sequence labeling

- Structured perceptron
  - A general algorithm for structured prediction problems such as sequence labeling
- The Argmax problem
  - Efficient argmax for sequences with Viterbi algorithm, given some assumptions on feature structure
  - A more general solution: Integer Linear Programming
- Loss-augmented structured prediction
  - Training algorithm
  - Loss-augmented argmax