POS tagging

CMSC 723 / LING 723 / INST 725

Marine Carpuat

POS tagging
Sequence labeling with the perceptron

Sequence labeling problem

- Input:
  - sequence of tokens $x = [x_1 \dots x_L]$
  - Variable length L
- Output (aka label):
  - sequence of tags $y = [y_1 \dots y_L]$
  - # tags = K
  - Size of output space?

Structured Perceptron

- Perceptron algorithm can be used for sequence labeling
- But there are challenges
  - How to compute argmax efficiently?
  - What are appropriate features?
- Approach: leverage structure of output space

Solving the argmax problem for sequences with dynamic programming

$x =$ " monsters eat tasty bunnies "

$y =$ noun verb adj noun

- Efficient algorithms possible if **the feature function decomposes over the input**

- This holds for unary and markov features used for POS tagging

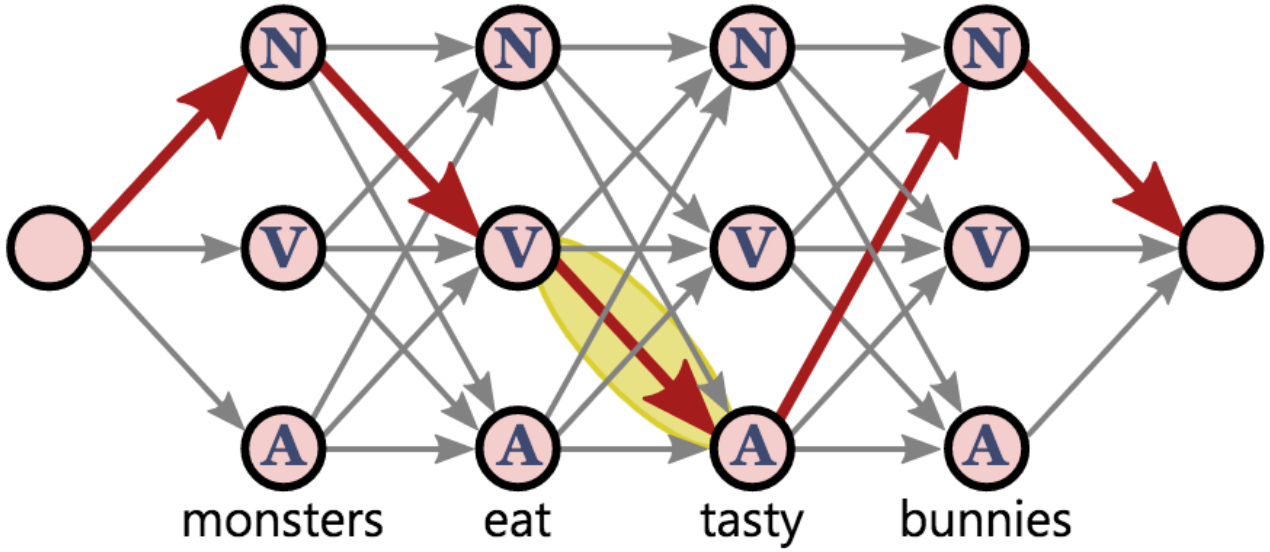
Feature functions for sequence labeling

$\boldsymbol{x}$ = " monsters eat tasty bunnies "

$\boldsymbol{y}$ = noun verb adj noun

- Standard features of POS tagging
  - **Unary features:** # times word w has been labeled with tag l for all words w and all tags l
  - **Markov features:** # times tag l is adjacent to tag l' in output for all tags l and l'
- Size of feature representation is constant wrt input length

Solving the argmax problem for sequences



- Trellis sequence labeling
  - Any path represents a labeling of input sentence
  - Gold standard path in red
  - Each edge receives a weight such that adding weights along the path corresponds to score for input/ouput configuration
- Any max-weight max-weight path algorithm can find the argmax
  - e.g. Viterbi algorithm $O(LK^2)$

Defining weights of edge in treillis

Unary features at position l together with Markov features that end at position l

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{w} \cdot \sum_{l=1}^{L} \boldsymbol{\phi}_l(\boldsymbol{x}, \boldsymbol{y}) \qquad \text{decomposition of structure} \qquad (17.35)$$

$$= \sum_{l=1}^{L} \boldsymbol{w} \cdot \boldsymbol{\phi}_l(\boldsymbol{x}, \boldsymbol{y}) \qquad \text{associative law} \qquad (17.36)$$

- Weight of edge that goes from time l-1 to time l, and transitions from y to y'

$$\boldsymbol{w} \cdot \boldsymbol{\phi}_l(\boldsymbol{x}, \cdots \circ y \circ y')$$

Dynamic program

- Define: the score of best possible output prefix up to and including position l that labels the l-th word with label k

$$\alpha_{l,k} = \max_{\hat{\boldsymbol{y}}_{1:l-1}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

- With decomposable features, alphas can be computed recursively

$$\alpha_{l+1,k} = \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \right]$$

$$\alpha_{0,k} = 0 \quad \forall k \tag{17.41}$$

$$\zeta_{0,k} = \emptyset \quad \forall k \tag{17.42}$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{\boldsymbol{y}}_{1:l}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \tag{17.43}$$

separate score of prefix from score of position l+1

$$= \max_{\hat{\boldsymbol{y}}_{1:l}} \boldsymbol{w} \cdot \left(\boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \right) \tag{17.44}$$

distributive law over dot products

$$= \max_{\hat{\boldsymbol{y}}_{1:l}} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k) \right] \tag{17.45}$$

separate out final label from prefix, call it k'

$$= \max_{\hat{\boldsymbol{y}}_{1:l-1}} \max_{k'} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k' \circ k) \right] \tag{17.46}$$

swap order of maxes, and last term doesn't depend on prefix

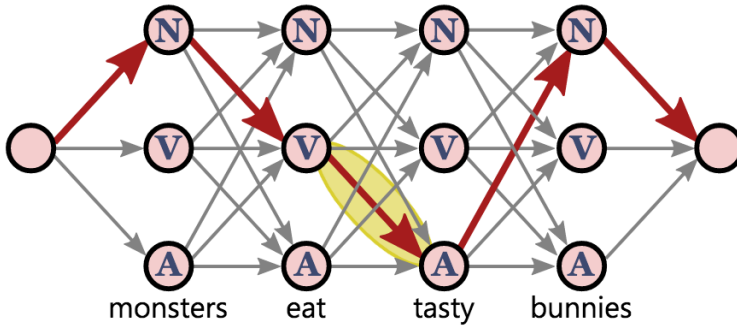
$$= \max_{k'} \left[\left[\max_{\hat{\boldsymbol{y}}_{1:l-1}} \boldsymbol{w} \cdot \boldsymbol{\phi}_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') \right] + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \right] \tag{17.47}$$

apply recursive definition

$$= \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle) \right] \tag{17.48}$$

Algorithm 42 ARGMAXFORSEQUENCES(x, w)

1: $L \leftarrow \text{LEN}(x)$
2: $\alpha_{l,k} \leftarrow o, \quad \zeta_{k,l} \leftarrow o, \quad \forall\, k = 1 \ldots K, \quad \forall l = 0 \ldots L$ // initialize variables
3: **for** $l = o \ldots L\text{-}1$ **do**
4: **for** $k = 1 \ldots K$ **do**
5: $\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + w \cdot \phi_{l+1}(x, \langle \ldots, k', k \rangle)\right]$ // recursion:
// here, $\phi_{l+1}(\ldots k', k \ldots)$ is the set of features associated with
// output position $l+1$ and two adjacent labels k' and k at that position
6: $\zeta_{l+1,k} \leftarrow$ the k' that achieves the maximum above // store backpointer
7: **end for**
8: **end for**
9: $y \leftarrow \langle o, o, \ldots, o \rangle$ // initialize predicted output to L-many zeros
10: $y_L \leftarrow \text{argmax}_k\, \alpha_{L,k}$ // extract highest scoring final label
11: **for** $l = L\text{-}1 \ldots 1$ **do**
12: $y_l \leftarrow \zeta_{l,y_{l+1}}$ // traceback ζ based on y_{l+1}
13: **end for**
14: **return** y // return predicted output



A more general approach for argmax Integer Linear Programming

- ILP: optimization problem of the form, for a fixed vector a

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

  - With integer constraints

- Pro: can leverage well-engineered solvers (e.g., Gurobi)
- Con: not always most efficient

POS tagging as ILP

- Markov features as binary indicator variables

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

  - Output sequence: y(z) obtained by reading off variables z
- Define a such that a.z is equal to score

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

- Enforcing constraints for well formed solutions

1. That all the zs are binary. That's easy: just say $z_{l,k',k} \in \{0,1\}$, for all l, k', k.
2. That for a given position l, there is exactly one active z. We can do this with an equality constraint: $\sum_k \sum_{k'} z_{l,k',k} = 1$ for all l.
3. That the zs are internally consistent: if the label at position 5 is supposed to be "noun" then both $z_{5,.,.}$ *and* $z_{6,.,.}$ need to agree on this. We can do this as: $\sum_{k'} z_{l,k',k} = \sum_{k''} z_{l+1,k,k''}$ for all l, k. Effectively what this is saying is that $z_{5,?,\text{verb}} = z_{6,\text{verb},?}$ where the "?" means "sum over all possibilities."

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 argmax
  - Hamming Loss

POS tagging

CMSC 723 / LING 723 / INST 725

Marine Carpuat