WestlakeNLP

Natural Language Processing

Yue Zhang
Westlake University

西湖大學
WESTLAKE UNIVERSITY

WestlakeNLP

Chapter 15

Neural Structured Prediction

Contents

Contents

Structure Prediction

- Sequence labelling (Chapter 7, 8)

Sentence	**POS tag sequence**
Jamie went to the shop yesterday .	*NNP VBD TO DT NN AD .*
What would you like to eat ?	*WP MD PRP VB TO VB .*
Tim is talking with Mary .	*NNP VBZ VBG IN NNP .*
I really appreciate it .	*PRP RB VBP PRP .*
John is a famous athlete .	*NNP VBZ DT JJ NN .*

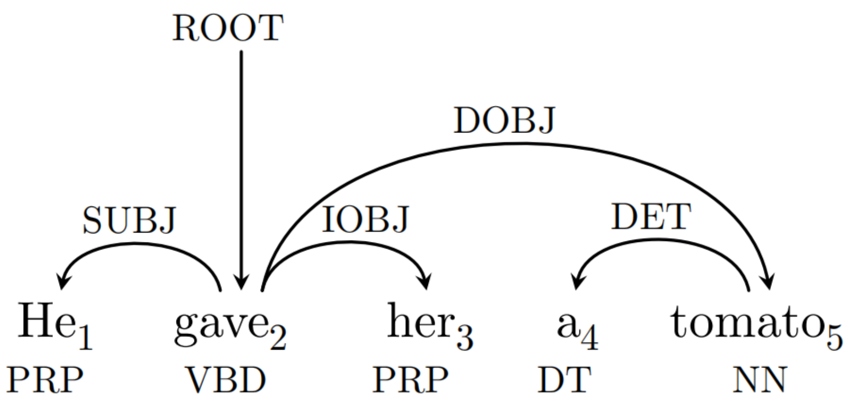
- Sequence Segmentation (Chapter 9)

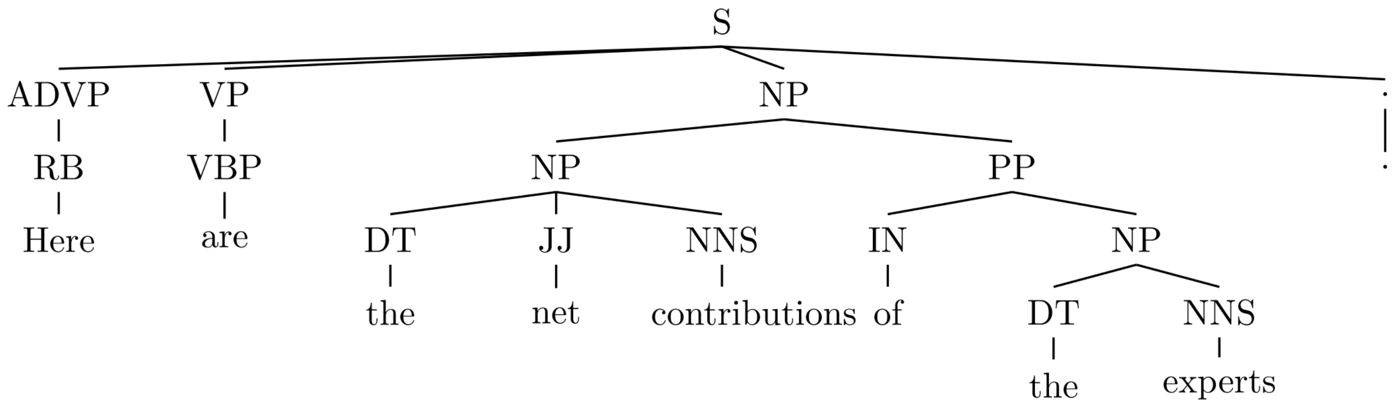
Word segmentation	其中国外企业	其中 国外 企业
	中国外企业务	中国 外企 业务

Input	**Output**
Michael Jordan is a Professor	[*Michael Jordan*]$_{\text{PER}}$ *is a Professor*
at University of Berkeley,	at [*University of Berkeley*]$_{\text{ORG}}$,
located near Silicon Valley, USA.	*located near* [*silicon valley*]$_{\text{LOC}}$, [*USA*]$_{\text{GPE}}$.
Mary went to Chicago to meet	[*Mary*]$_{\text{PER}}$ *went to* [*Chicago*]$_{\text{LOC}}$
her boyfriend John Smith.	*to meet her boyfriend* [*John Smith*]$_{\text{PER}}$.

Structure Prediction

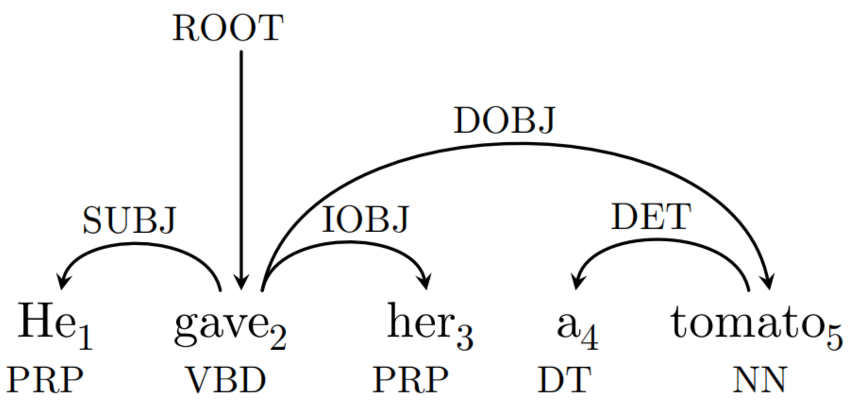
- Sequence labelling (Chapter 7, 8)
- Sequence Segmentation (Chapter 9)
- Tree Structure Prediction (Chapter 10, 11)

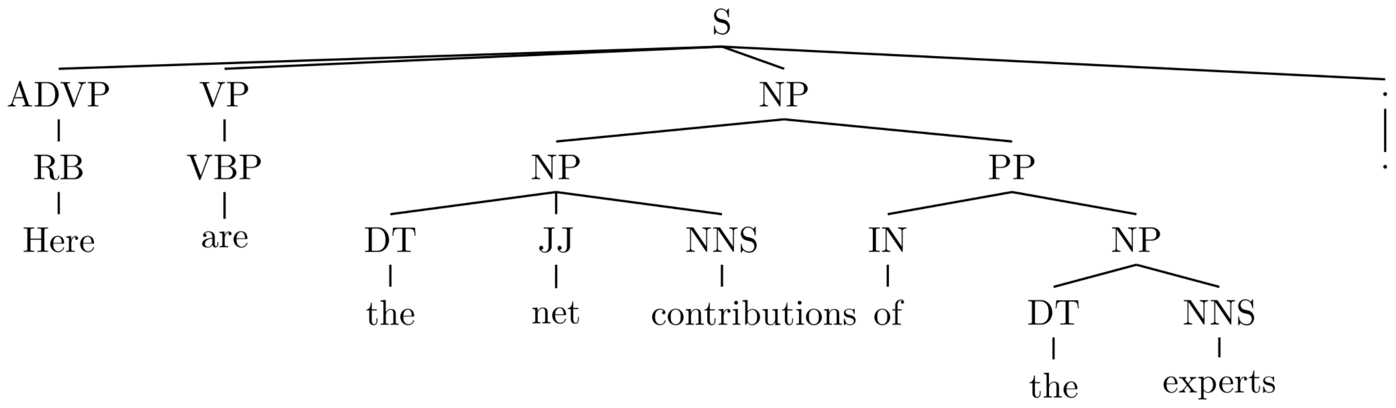




Structure Prediction

- Sequence labelling (Chapter 7, 8)
- Sequence Segmentation (Chapter 9)
- Tree Structure Prediction (Chapter 10, 11)





Graph-based models (Chapters 7 – 10) vs Transition-based models (Chapter 11)
Local model vs Global model

Local Graph-Based Models

WestlakeNLP

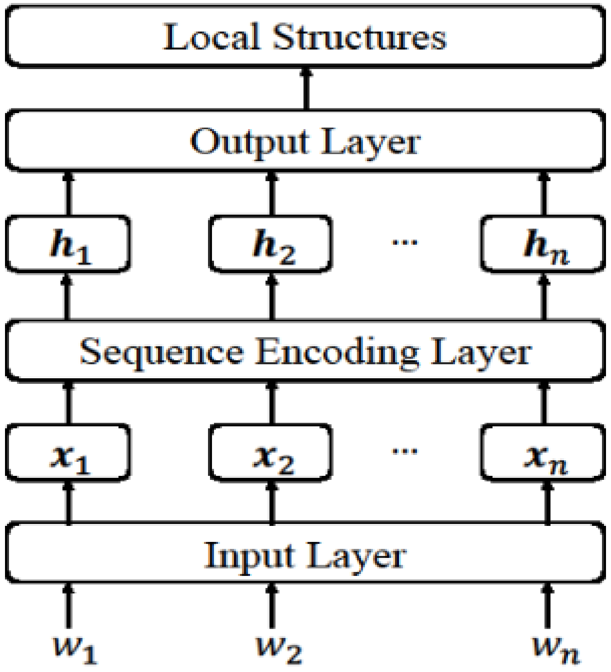
- Comparison with linear models
  - No complex discrete features
  - No dynamic program

- Overview

  Input: $W_{1:n} = w_1, w_2, \dots, w_n$

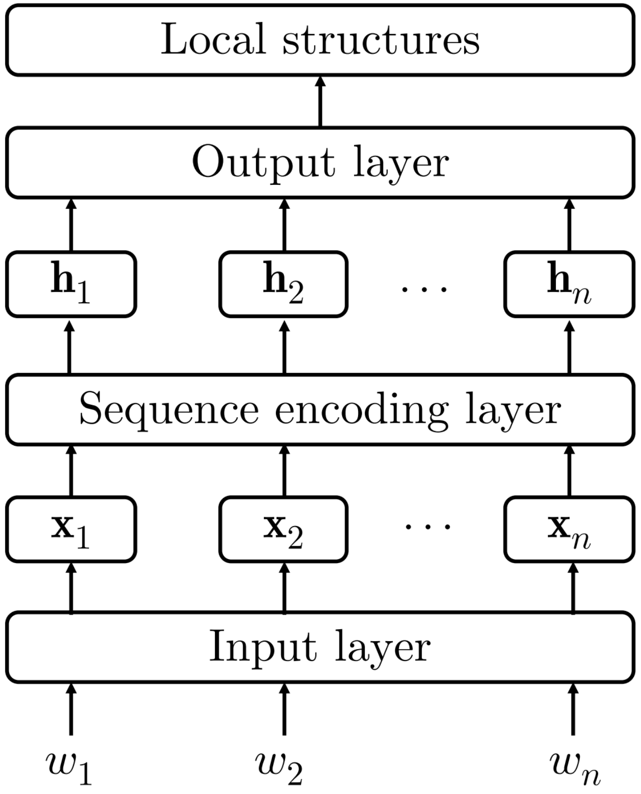
  Embeddings: $\boldsymbol{X}_{1:n} = \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$

  Hidden: $\boldsymbol{H}_{1:n} = \boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_n$

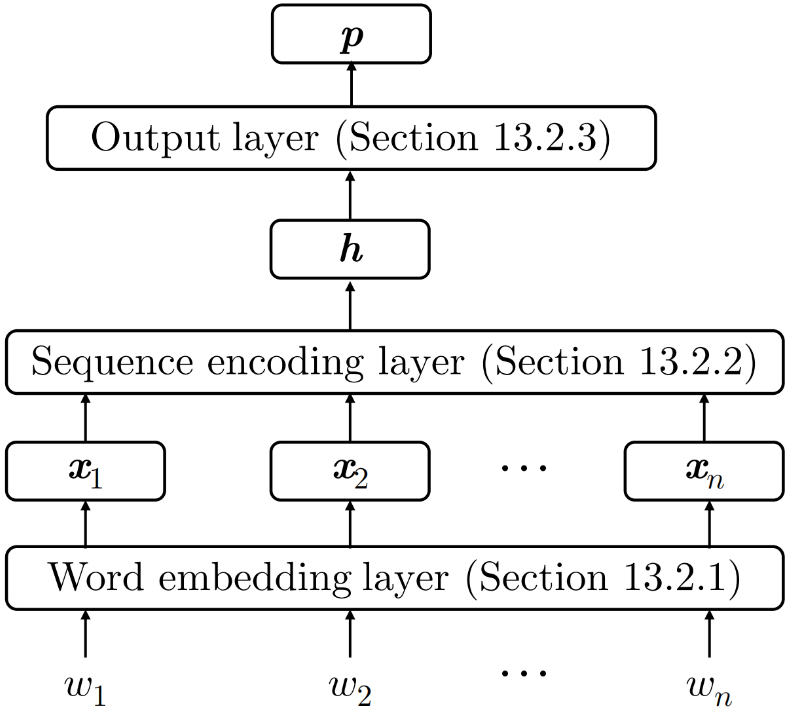


Local Graph-Based Models

- Structure prediction *vs.* Classification



Structure prediction



Classification

Contents

Local Graph-Based Models

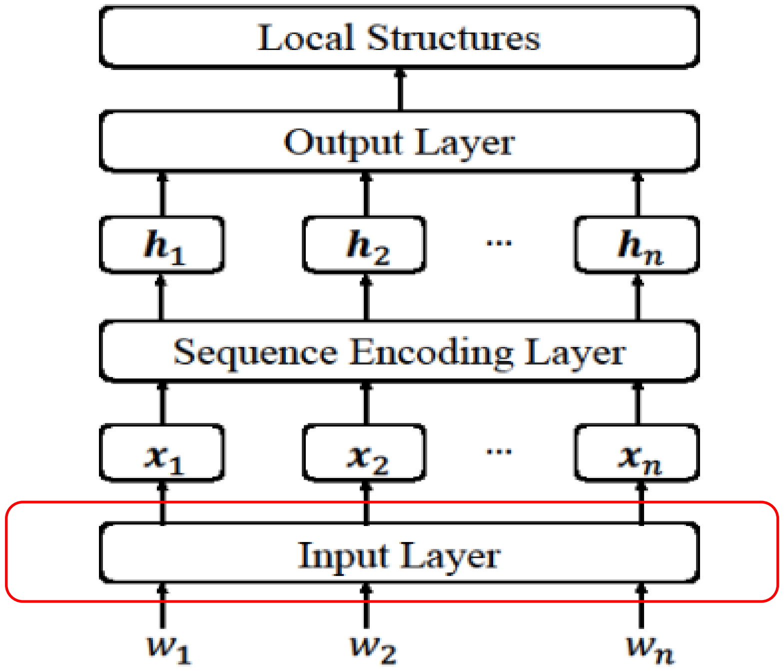
- Sequence labelling
  - Task:

$$W_{1:n} \rightarrow T_{1:n} = t_1, t_2, \dots, t_n$$

  - Input layer

$$\boldsymbol{x}_i = emb(w_i)$$

  - OOV words?
    - add a special token <OOV> to represent OOV words.
    - during training, randomly flip infrequent words into <OOV>.



Local Graph-Based Models

WestlakeNLP

- Sequence labelling
  - Task:

$$W_{1:n} \rightarrow T_{1:n} = t_1, t_2, \dots, t_n$$

  - Input layer

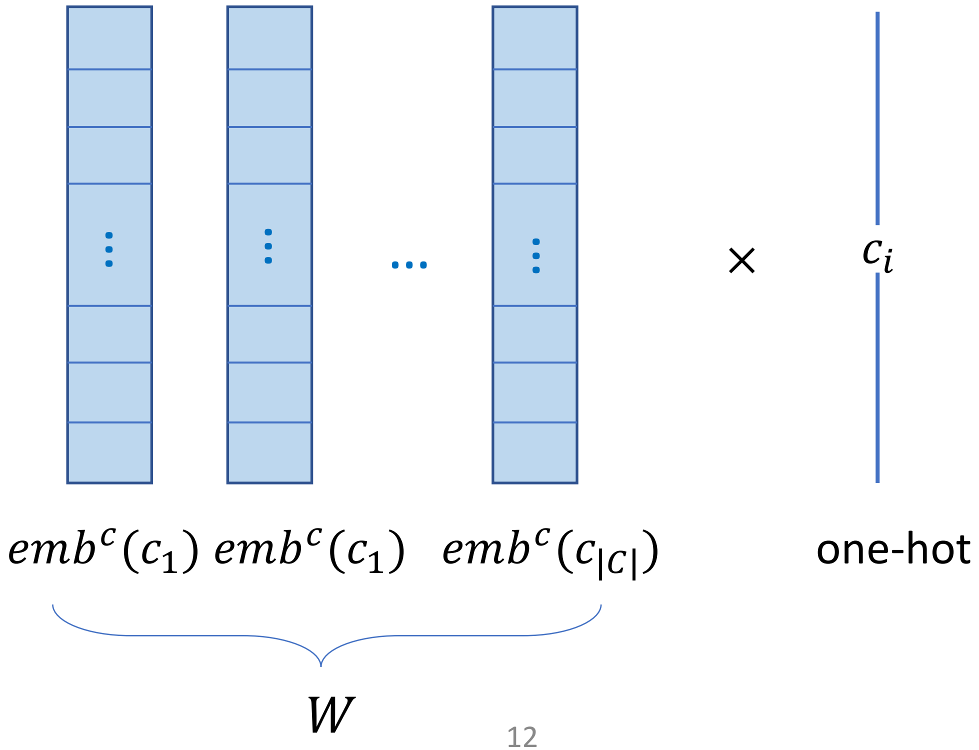
$$w_i = C_{1:|w_i|} = c^i_1, c^i_2, \dots, c^i_{|w_i|}$$

$$\mathbf{x}^c_{w_i} = [emb^c(c^i_1) \dots emb^c(c^i_{|w_i|})]$$

$$chr(w_i) = Encoder\left(\mathbf{x}^c_{w_i}\right)$$

$$\mathbf{x}_i = chr(w_i) \oplus emb(w_i)$$

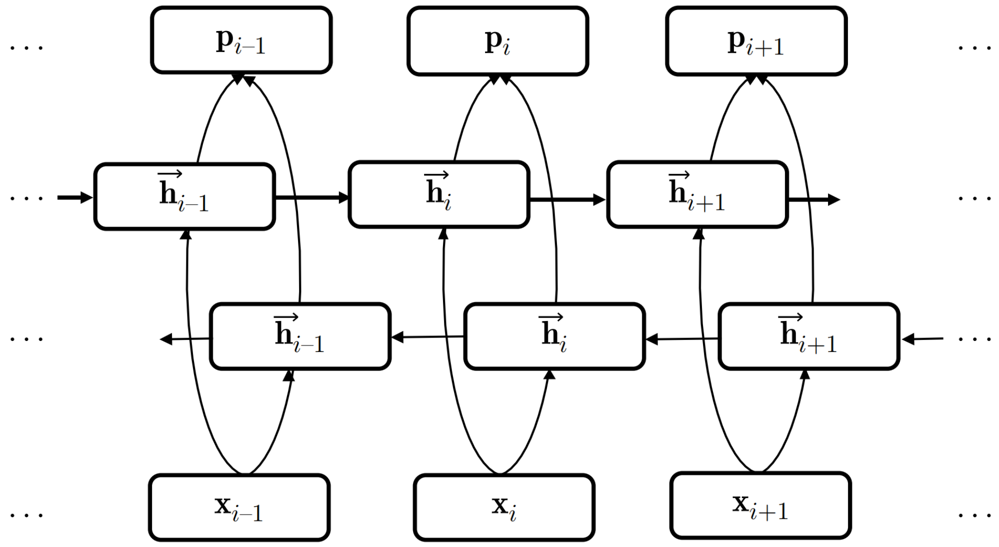
$$emb^c(\mathrm{c}) = W \cdot \text{one}-\text{hot}(c)$$

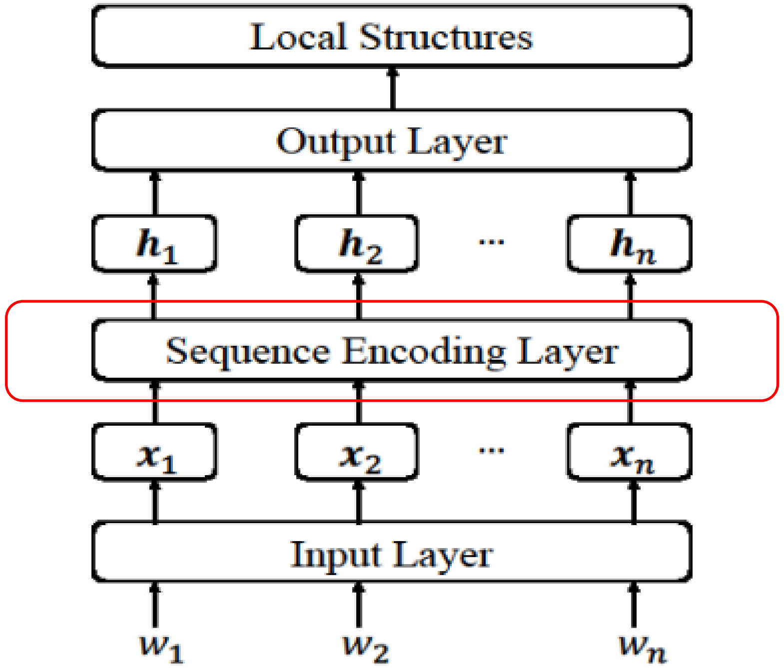


Local Graph-Based Models

WestlakeNLP

- Sequence labelling
  - Sequence representation layer





$$\mathbf{H}_{1:n} = BiLSTM(\boldsymbol{X}_{1:n})$$

  - Can stack multi layers.

Local Graph-Based Models

- Sequence labelling
  - Each $\boldsymbol{x}_i$ is classified using $\boldsymbol{h}_i$
  - There are $|L|$ labels.
  - Output layer

$$\mathbf{o}_i = \mathbf{W}\mathbf{h}_i + \mathbf{b} \qquad \mathbf{p}_i = softmax(\mathbf{o}_i)$$

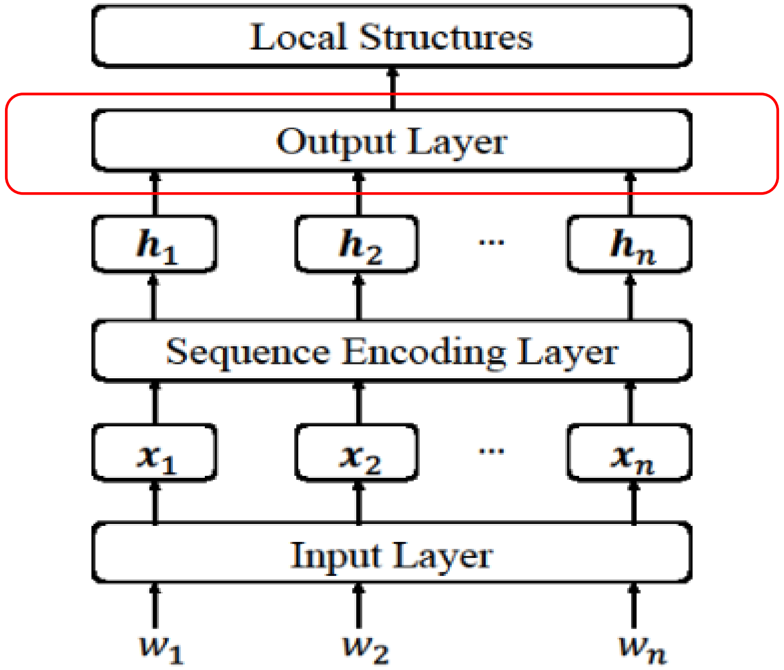
$\mathbf{o}_i$ and $\mathbf{p}_i$ have $|L|$ dimensions

$\mathbf{p}_i[j]$ denotes $p(t_i = l_j | w_{1:n})$

  - Training

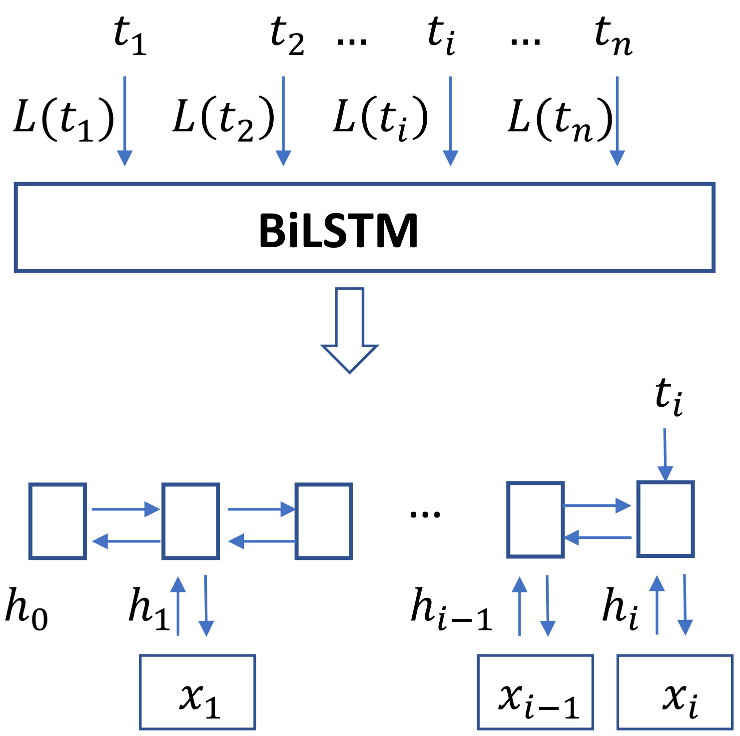
$$D = \{(W_i, T_i)\}|_{i=1}^{N}$$

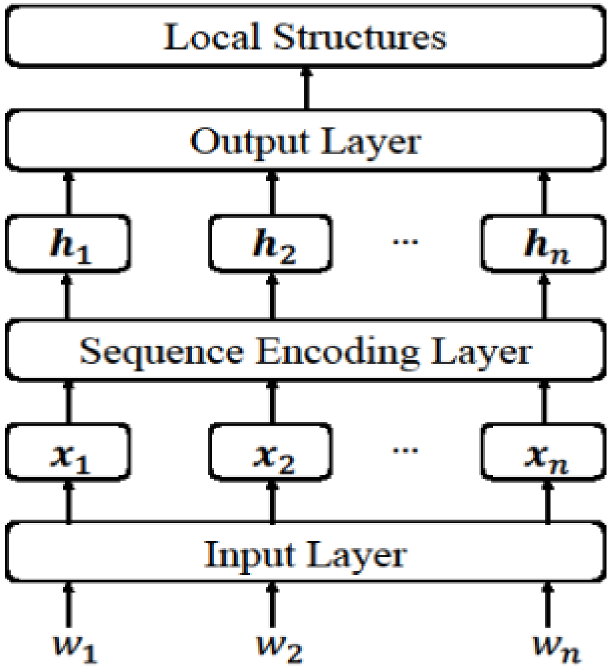
$$L = -\sum_{i}^{N} \sum_{j=1}^{|W_i|} \log\left(\mathbf{p}_j^i[t_j^i]\right)$$



Local Graph-Based Models

- There is a loss at every t_i (i ∈ $[1, \ldots, n]$).
- Gradient Propagation for RNN-based models





- Gradients from each label are accumulated.

Contents

WestlakeNLP

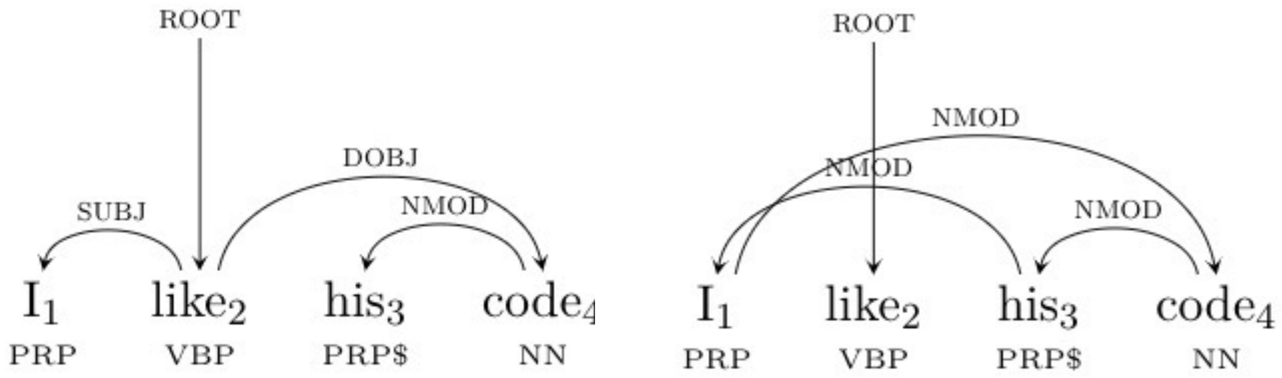
Local Graph-Based Models

- Dependency parsing
  - Input

$$S = (w_1, t_1), (w_2, t_2), \ldots, (w_n, t_n)$$

  - Output

$$o = \{(i, h_i, l_i)\}|_{i=1}^{n}$$



- $w_0 = ROOT$ pseudo root

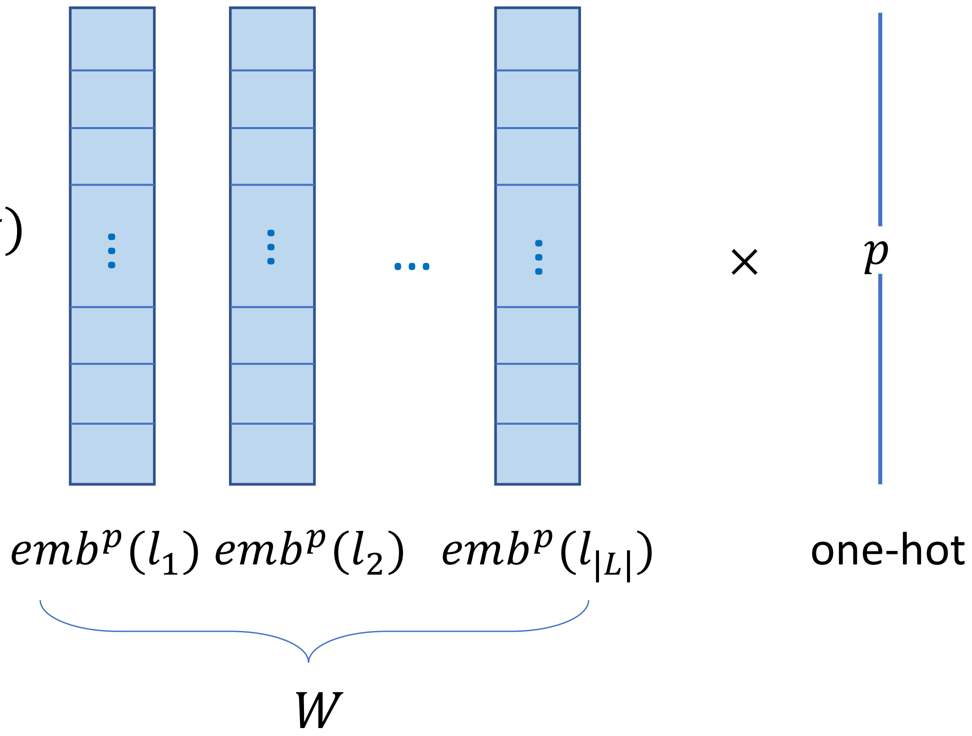
Local Graph-Based Models

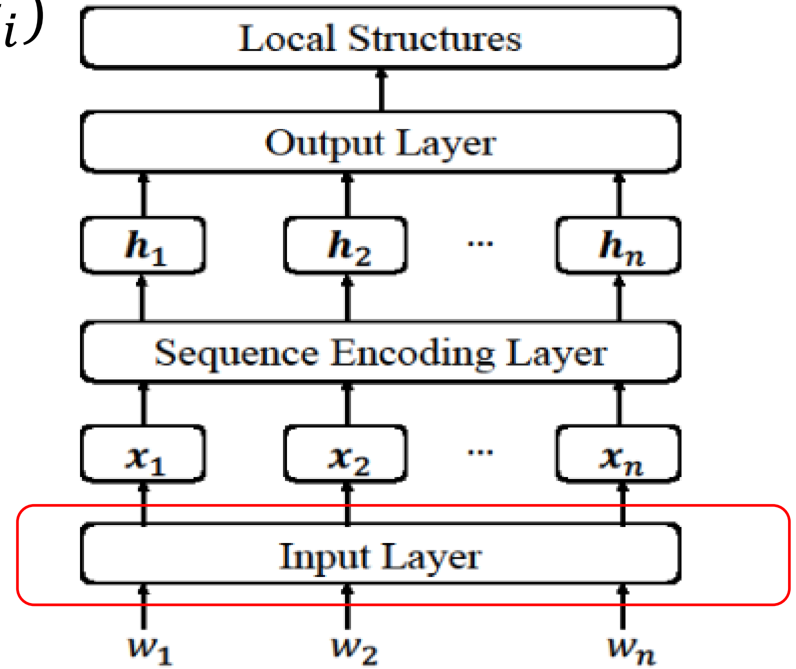
- Dependency parsing
  - Input layer

$$\mathbf{x}_i = emb(w_i) \oplus chr(w_i) \oplus emb^p(t_i)$$

  - POS embedding

$$emb^p(\mathrm{t}) = W \cdot \text{one-hot}(t)$$

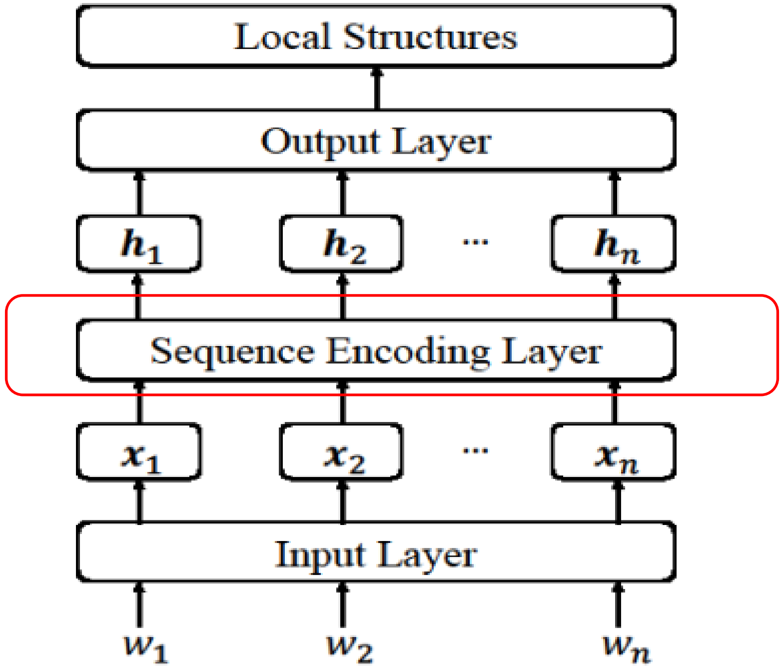




Local Graph-Based Models

- Dependency parsing
  - Sequence encoding layer

$$\mathbf{H}_{1:n} = BiLSTM(\boldsymbol{X}_{1:n})$$



Local Graph-Based Models

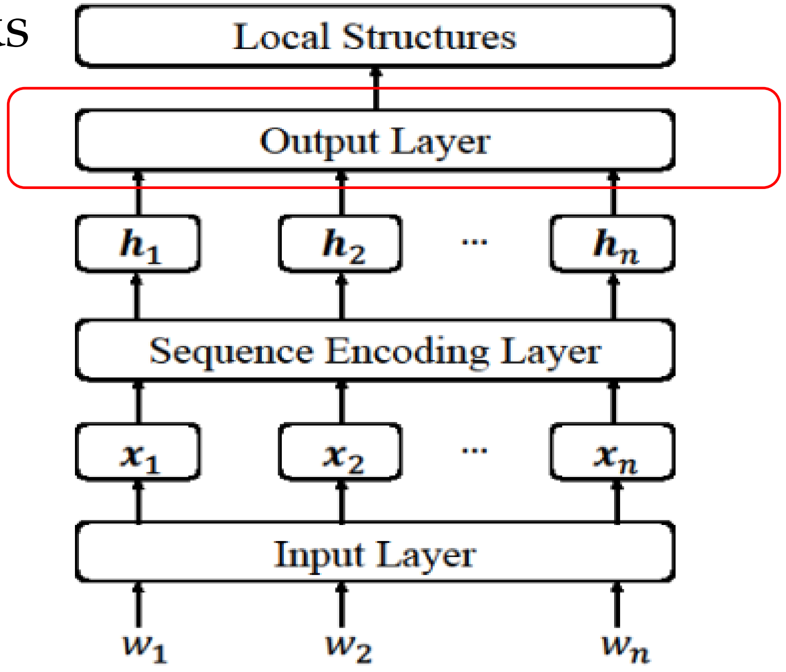
- Dependency parsing
  - Output layer
    - Structure assignment (each word i looks for a head j ,including <ROOT>=0)

$$s_{i,j} = \mathbf{h}_i^T \mathbf{U} \mathbf{h}_j + \mathbf{v}^{\mathbf{T}}([\mathbf{h}_i; \mathbf{h}_j])$$

$$\mathbf{o}_i^{arc} = \langle s_{i,1}, s_{i,2}, \ldots, s_{i,n} \rangle$$

$$\mathbf{p}_i^{arc} = softmax(\mathbf{o}_i^{arc})$$

$$h_i = \underset{h}{\operatorname{argmax}} \mathbf{p}_i^{arc}[h]$$



- $\boldsymbol{o}, \boldsymbol{p}$ has $n+1$ dimensions, and $\boldsymbol{p}_i^{arc}[j] = P(w_j \; is \; head \; of \; w_i)$

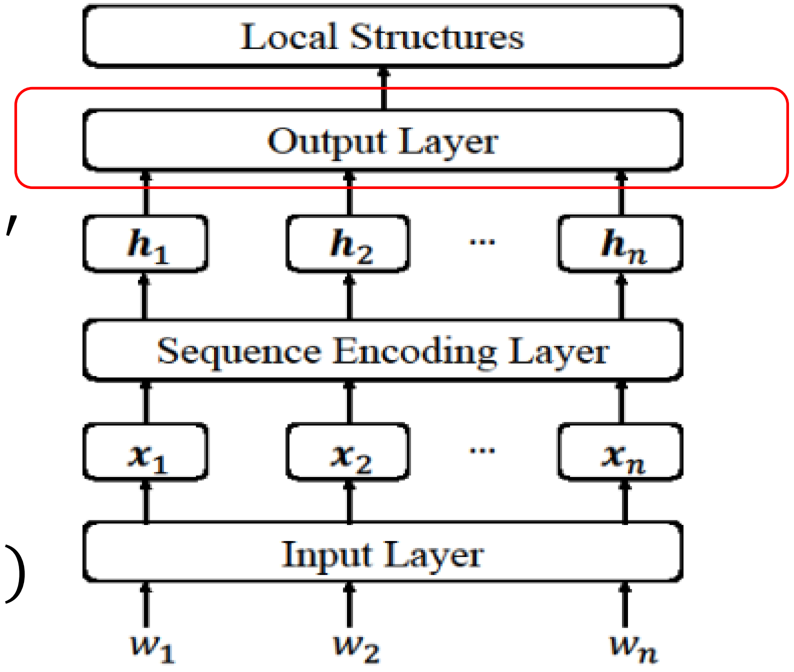
Local Graph-Based Models

WestlakeNLP

- Dependency parsing
  - Output layer
    - Label assignment

$$\mathbf{o}_i^{label} = \mathbf{h}_i^T \mathbf{U}' \mathbf{h}_{h_i} + \mathbf{V}'([\mathbf{h}_i; \mathbf{h}_{h_i}]) + \mathbf{b}'$$

$$\mathbf{p}_i^{label} = softmax(\mathbf{o}_i^{label})$$



- $\mathbf{p}_i^{label}$[j] $= p(arc\ i \leftarrow h_i$ has label l_j)
- Results in invalid tree?
- Use a dynamic program.

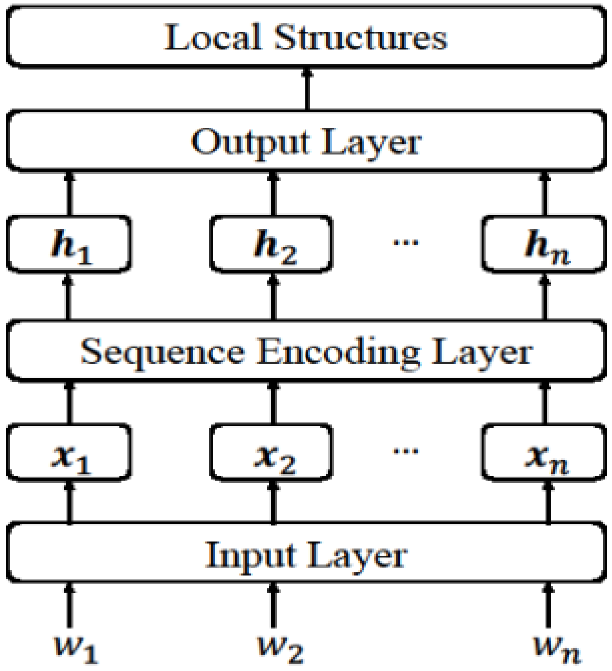
Local Graph-Based Models

- Dependency parsing
  - Training

$$D = \{(S_i, T_i)\}|_{i=1}^{N} \qquad T_i = \{(j, h_j^i, l_j^i)\}|_{j=1}^{|S_i|}$$

$$L = -\sum_{i}^{N}\sum_{j=1}^{|W_i|}\left(\log\left(\left(\mathbf{p}_j^i\right)^{arc}[h_j^i]\right) + \log\left(\left(\mathbf{p}_j^i\right)^{label}[l_j^i]\right)\right)$$

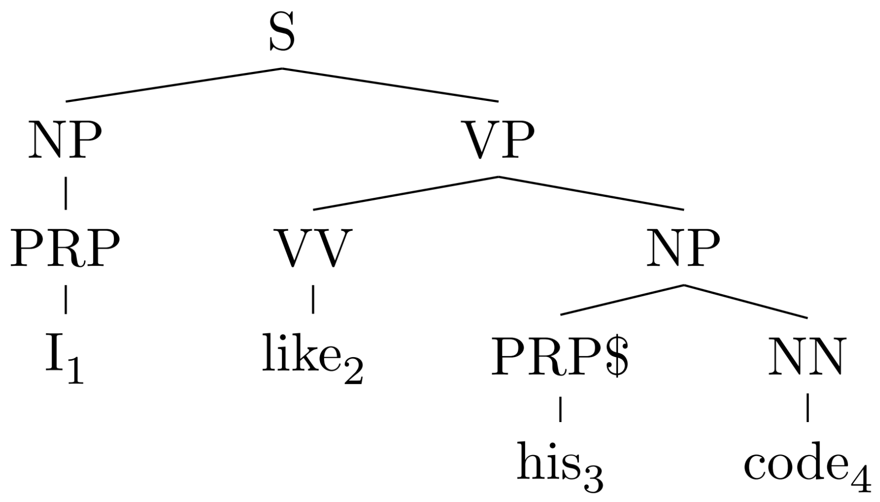
  - Loss from every arc accumulates via $\boldsymbol{h}_i, \boldsymbol{h}_j$
  - Loss from every arc label accumulates via $\boldsymbol{h}_i, \boldsymbol{h}_j$



Contents

Local Graph-Based Models

- Constituent parsing
  - Task
    - Input $S = (w_1, t_1), (w_2, t_2), \dots, (w_n, t_n)$
    - Output $T = \{(b, e, c)\}, 1 \leq b \leq e \leq n, c = [b, e]$
    - Local model to classify each span.



Local Graph-Based Models

- Constituent parsing
  - Input layer

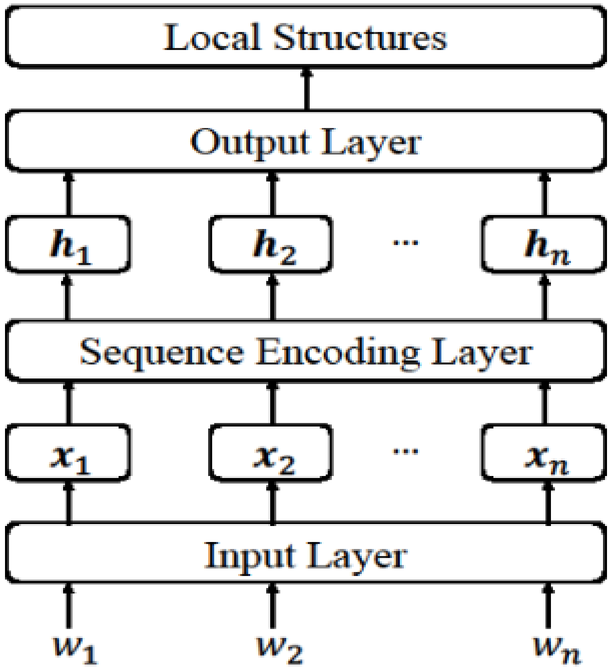
$$chr(w_i) = \tanh\left(\mathbf{W}_e^{char}\mathbf{ch}_i^l + \mathbf{W}_r^{char}\mathbf{ch}_i^r + \mathbf{b}^{char}\right)$$
$$\mathbf{x}_i = emb(w_i) \oplus chr(w_i) \oplus emb^p(t_i)$$

  - Sequence encoding layer (stacked bi-LSTM)

$$\mathbf{s}[b, e] = \mathbf{h}_\mathrm{b} \oplus \mathbf{h}_\mathrm{e}$$

  - Output layer

$$\mathbf{h}[b, e] = \tanh\left(\mathbf{W}^h\mathbf{s}[b, e] + \mathbf{b}^h\right)$$
$$\mathbf{o}[b, e] = \mathbf{W}^o\mathbf{h}[b, e] + \mathbf{b}^o$$
$$\mathbf{p}[b, e] = softmax(\mathbf{o}[b, e])$$



Local Graph-Based Models

- Constituent parsing
  - Decoding (find structure first)

$$P(y=1|S,b,e)=\sum_{c,c\neq\phi}P(c|S,b,e)=1-P(\phi|S,b,e)$$

$$P(y=0|S,b,e)=P(\phi|S,b,e)$$

  - Rule (disregarding consistent labels)

$$W_{b:e}\rightarrow W_{b:b'-1}W_{b':e}$$

$$P(r\mid S,b,e)=P(y=1\mid S,b,b'-1)P(y=1\mid S,b',e)$$

  - CKY

Local Graph-Based Models

WestlakeNLP

- Constituent parsing
  - Training

$$D = \{(S_i, T_i)\}|_{i=1}^{N}$$

$$S_i = \left(w_1^i, t_1^i\right), \dots, \left(w_{|S_i|}^i, t_{|S_i|}^i\right)$$

$$T_i = \left\{\langle b_k^i, e_k^i, l_{b_k^i, e_k^i}^i \rangle\right\} |_{k=1}^{|T_i|}$$

$$L = -\sum_{i=1}^{N} \sum_{1 \le b \le |S_i|, b \le e \le |S_i|} \log P(c|S, b, e)$$

$c = l_{b,e}\ if\ (b, e, c) \in T_i$ and ϕ otherwise.

Contents

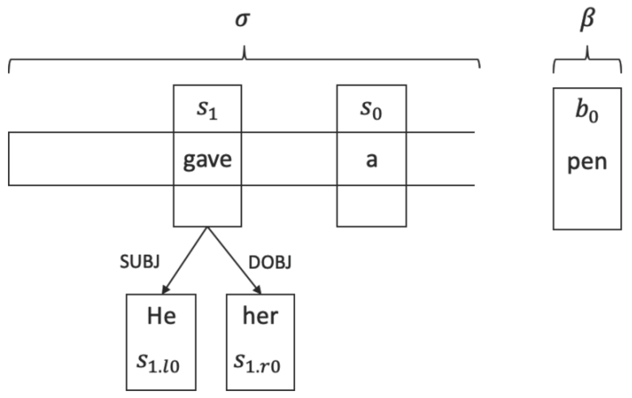
WestlakeNLP

Local Transition-Based Models

WestlakeNLP

- Use state-transitions to model output (Chapter 11)
- State transitions
  - *State* represents a partially constructed output
  - *Transition actions* represent incremental steps for building structures
- Example

$$s = (\sigma, \beta, A)$$
$$\sigma = [\ldots, s_1, s_0]$$
$$\beta = [b_0, b_1, \ldots]$$



$\sigma=[\text{gave, a}]$, $\beta=[\text{pen}]$ and $A=\{(\text{He} \overset{\text{SUBJ}}{\curvearrowleft} \text{gave}), (\text{gave} \overset{\text{DOBJ}}{\curvearrowright} \text{her})\}$

Local Transition-Based Models

Arc-standard Projective Dependency

- Example

Next action: SHIFT

He gave her a pen

Local Transition-Based Models

Arc-standard Projective Dependency

- Example

Next action: SHIFT

Stack		Buffer
He		gave her a pen

Local Transition-Based Models

Arc-standard Projective Dependency

- Example

  Next action: LEFT-ARC-SUBJ

Stack	Buffer
He gave	her a pen

Local Transition-Based Models

WestlakeNLP

Arc-standard Projective Dependency

- Example

  Next action: SHIFT

Local Transition-Based Models

Arc-standard Projective Dependency

- Example

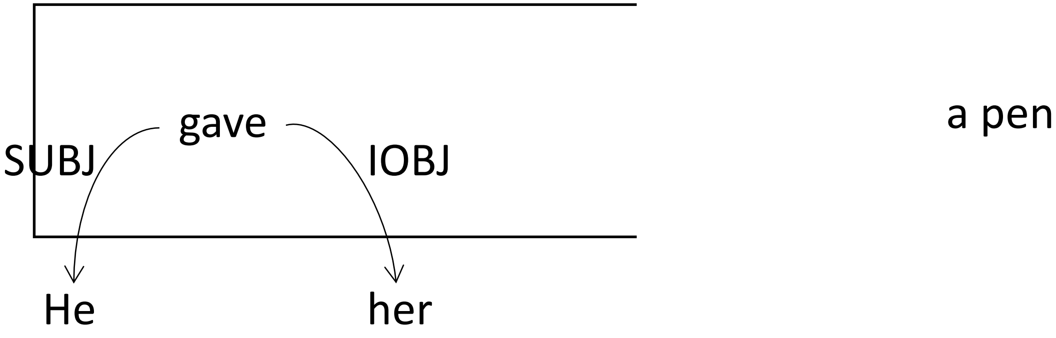
Next action: RIGHT-ARC-IOBJ

Local Transition-Based Models

Arc-standard Projective Dependency

- Example

  Next action: SHIFT

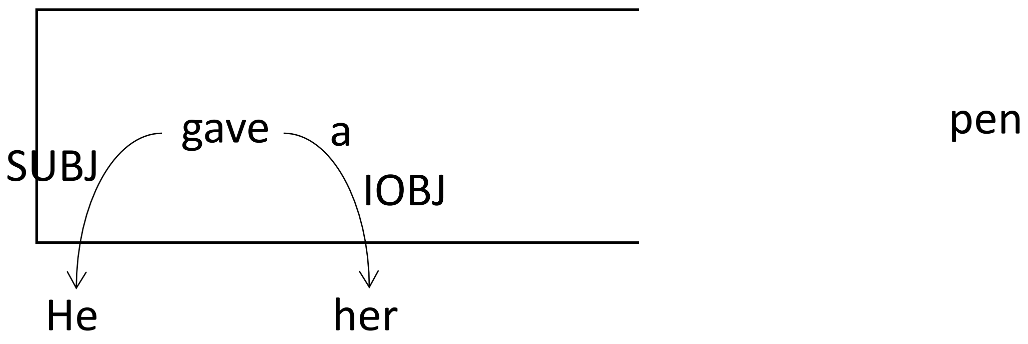


Local Transition-Based Models

Arc-standard Projective Dependency

- Example

  Next action: SHIFT

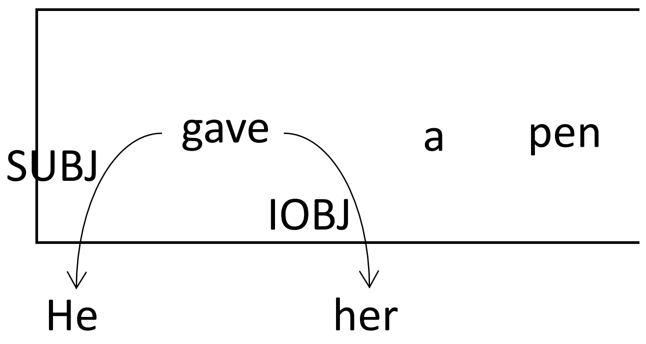


Local Transition-Based Models

Arc-standard Projective Dependency

- Example

  Next action: LEFT-ARC-DET

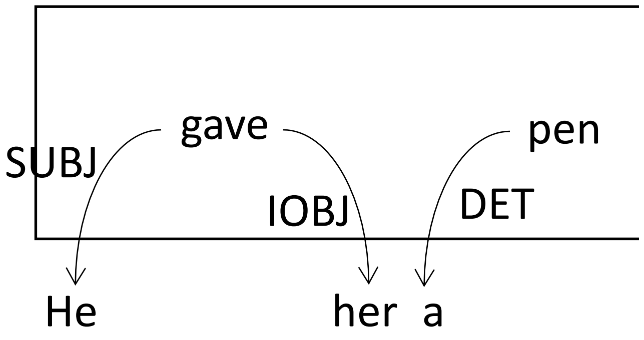


Local Transition-Based Models

Arc-standard Projective Dependency

- Example

  Next action: RIGHT-ARC-DOBJ

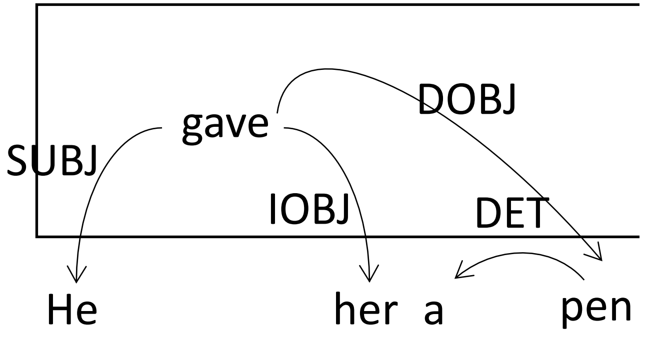


Local Transition-Based Models

Arc-standard Projective Dependency

- Example

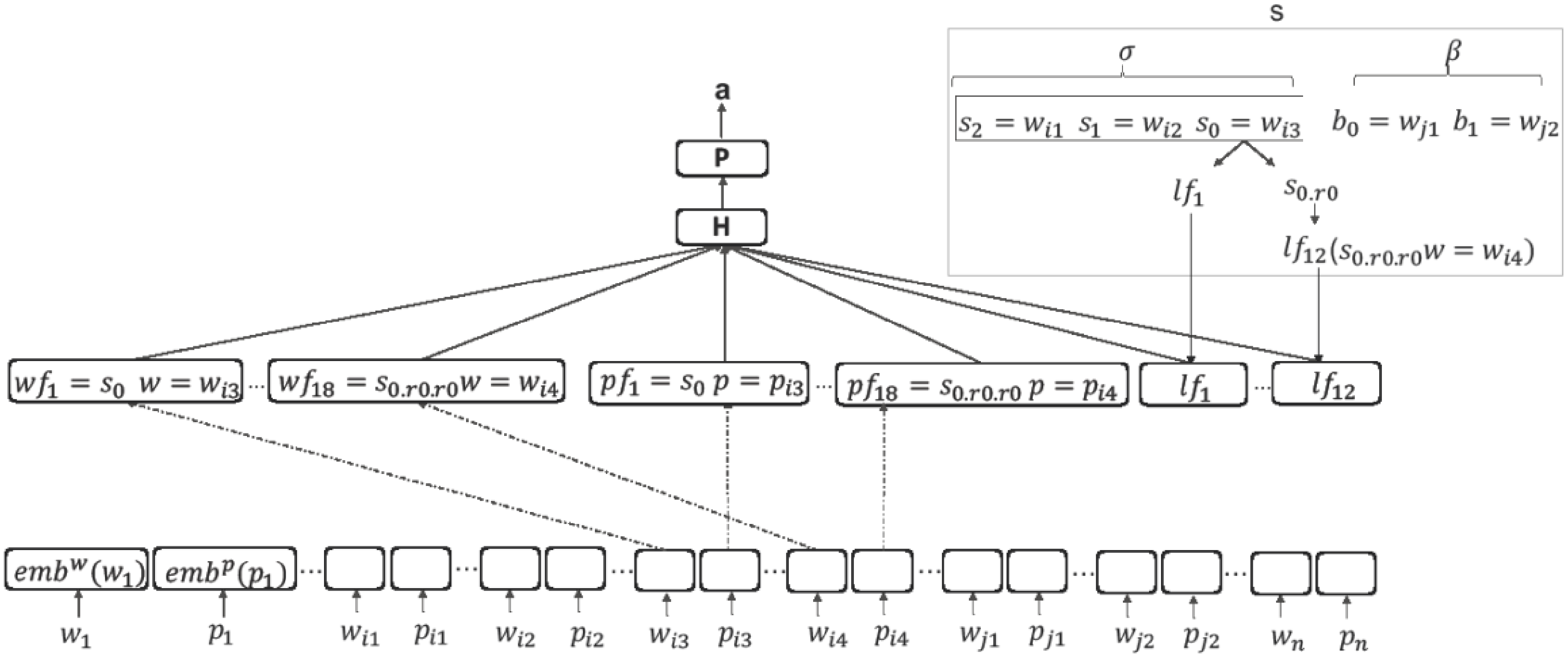
Next action: END



Contents

Local Transition-Based Models

- Model 1 -- directly takes embedding features.



Local Transition-Based Models

- Model 1 -- MLP classifier

Type	Feature template
word features (n_w=18)	$s_0w, s_1w, s_2w, b_0w, b_1w, b_2w$ $s_{0.l0}w, s_{0.l1}w, s_{1.l0}w, s_{1.l1}w, s_{0.r0}w, s_{0.r1}w, s_{1.r0}w, s_{1.r1}w$ $s_{0.l0.l0}w, s_{0.r0.l0}w, s_{0.l0.r0}w, s_{0.r0.r0}w$
POS features (n_p=18)	$s_0p, s_1p, s_2p, b_0p, b_1p, b_2p$ $s_{0.l0}p, s_{0.l1}p, s_{1.l0}p, s_{1.l1}p, s_{0.r0}p, s_{0.r1}p, s_{1.r0}p, s_{1.r1}p$ $s_{0.l0.l0}p, s_{0.r0.l0}p, s_{0.l0.r0}p, s_{0.r0.r0}p$
arc features (n_l=12)	$s_{0.l0}l, s_{0.l1}l, s_{1.l0}l, s_{1.l1}l, s_{0.r0}l, s_{0.r1}l, s_{1.r0}l, s_{1.r1}l$ $s_{0.l0.l0}l, s_{0.r0.l0}l, s_{0.l0.r0}l, s_{0.r0.r0}l$

$$\mathbf{X}^w = [e; m; b(wf_1) \dots emb(wf_{n_w})]$$
$$\mathbf{X}^p = [e; m; b^p(pf_1) \dots emb^p(pf_{n_p})]$$
$$\mathbf{X}^l = [e; m; b^l(lf_1) \dots emb^l(lf_{n_l})]$$
$$\mathbf{h} = f(\mathbf{W}^w\mathbf{X}^w + \mathbf{W}^p\mathbf{X}^p + \mathbf{W}^l\mathbf{X}^l + \mathbf{b}_h)$$
$$\mathbf{o} = \mathbf{W}^\mathbf{o}\mathbf{h} + \mathbf{b}^\mathbf{o}$$
$$\mathbf{p} = softmax(\mathbf{o})$$

Local Transition-Based Models

- Model 1
  - Training

$$D = \{(W_i, T_i)\}|_{i=1}^{N}$$

$$T_i = \langle (s_0^i, a_1^i), (s_1^i, a_2^i), \ldots, (s_{i|W_i|-2}^i, a_{2|W_i|-1}^i) \rangle$$

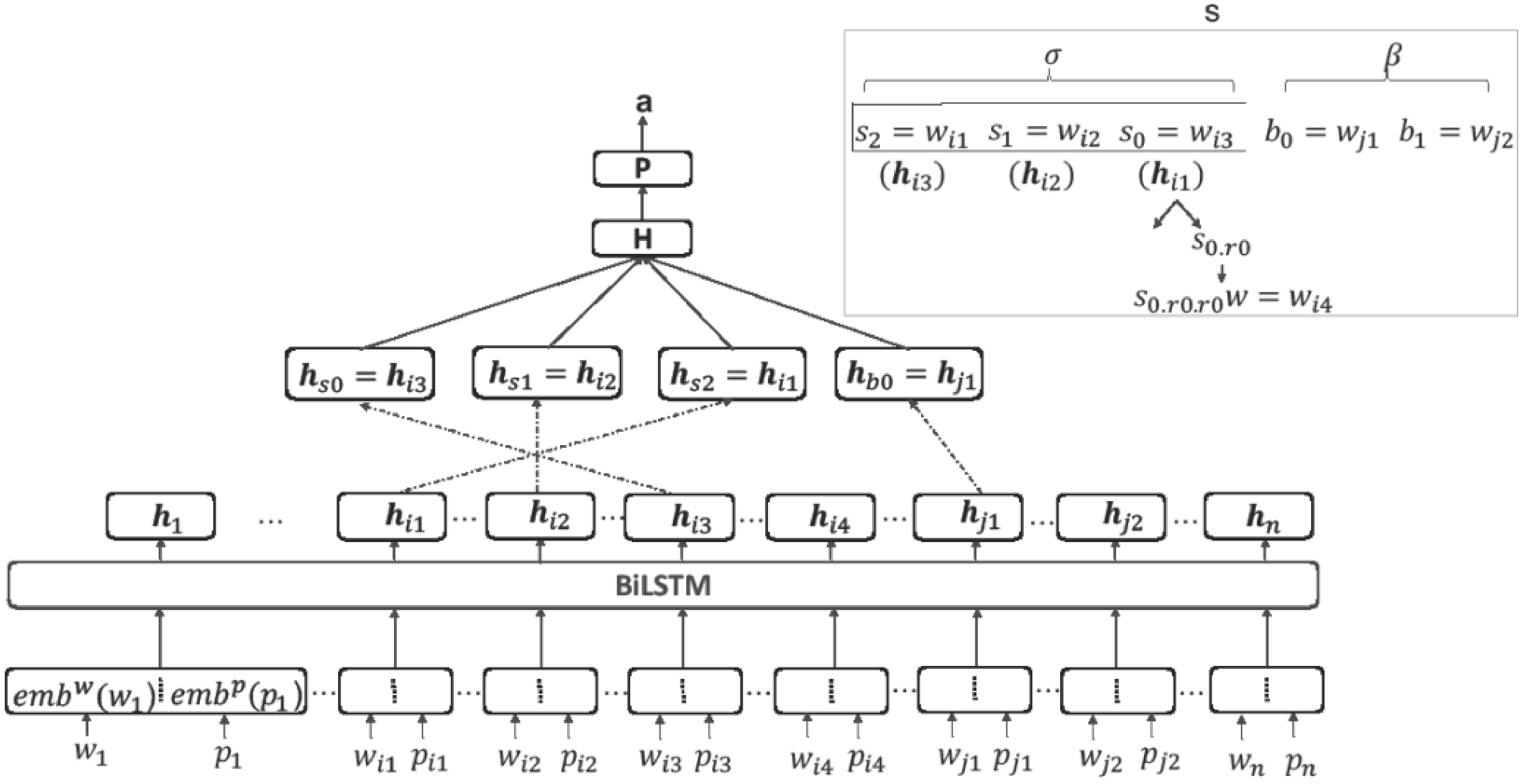
$$L = -\sum_{i=1}^{N} \sum_{j=1}^{2|W_i|-1} \log P(a_j^i | s_{j-1}^i)$$

Contents

WestlakeNLP

Local Transition-Based Models

- Model 2 – enrich input sequence features
  - Sequence encoding



Local Transition-Based Models

- Model 2 – reduce stack feature sources.

$$\mathbf{x}_i = emb(w_i) \oplus emb^p(t_i)$$

$$\mathbf{H}_{1:n} = BiLSTM(\boldsymbol{X}_{1:n})$$

$$\mathbf{h} = \mathbf{h}_{s_0} \oplus \mathbf{h}_{s_1} \oplus \mathbf{h}_{s_2} \oplus \mathbf{h}_0$$

$$\mathbf{o} = \mathbf{W}^o \tanh\left(\mathbf{W}^h \mathbf{h} + \mathbf{b}^h\right) + \mathbf{b}^o$$

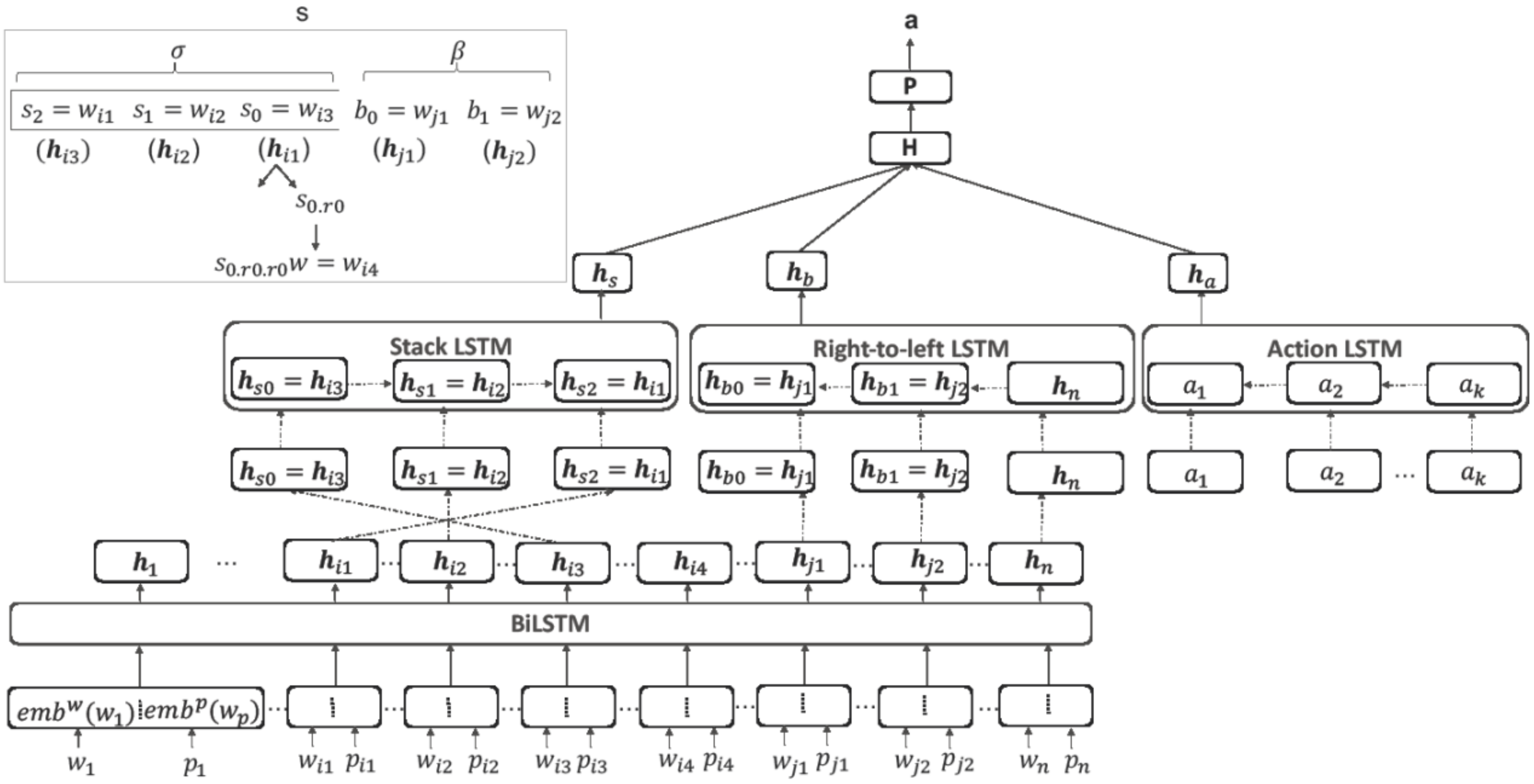
$$\mathbf{p} = softmax(\mathbf{o})$$

Contents

WestlakeNLP

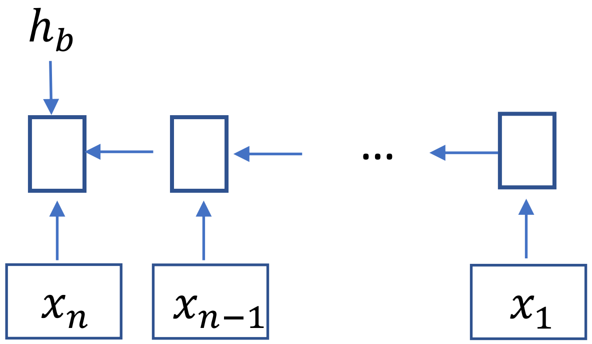
Local Transition-Based Models

- Model 3 –better represents the state

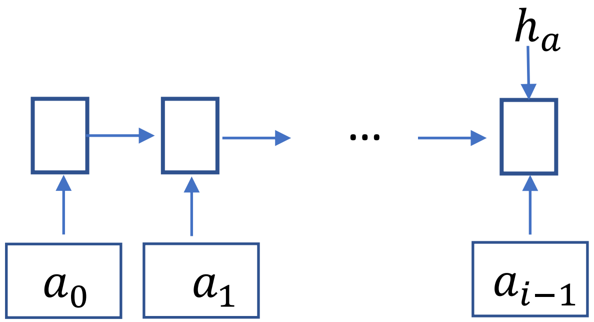


Local Transition-Based Models

- Model 3
  - Buffer LSTM

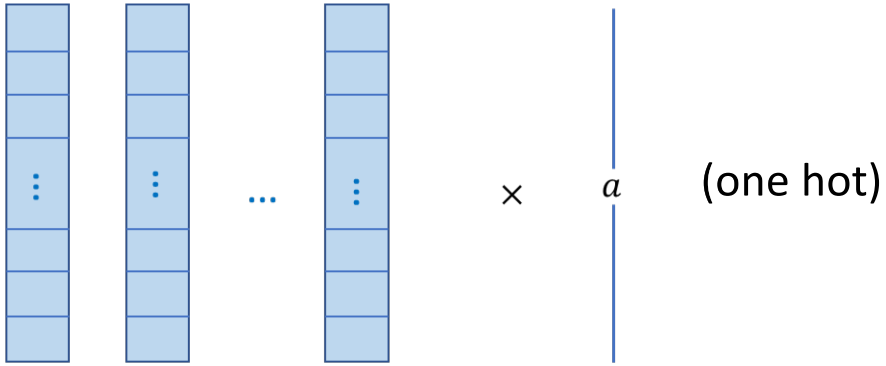


  - Action LSTM



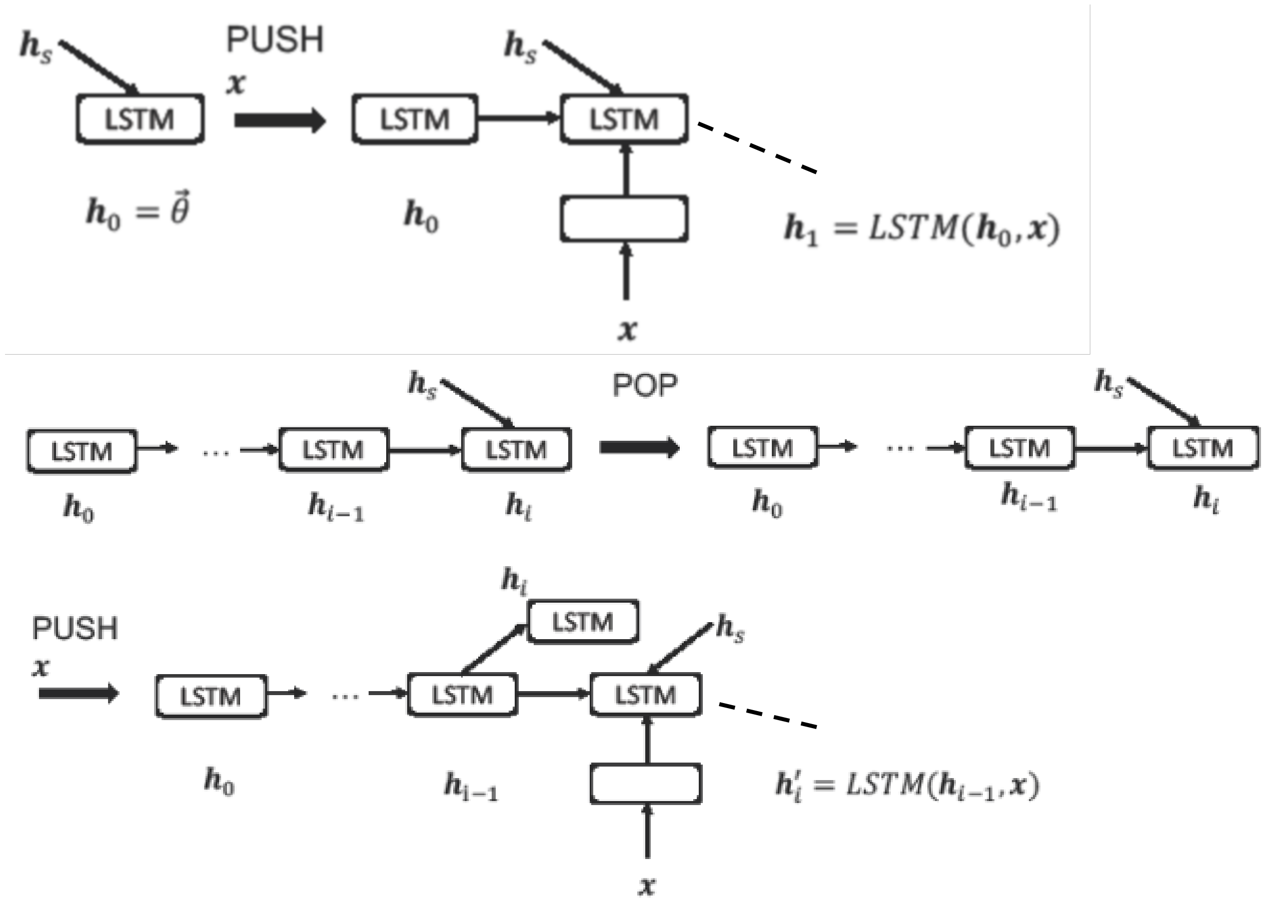
  - action embedding

    $emb(a) = W$



Local Transition-Based Models

- Model 3
  - Stack LSTM



Local Transition-Based Models

- Model 3
  - Further reduces stack features

$$\mathbf{h} = \mathbf{h}_{\boldsymbol{s}} \oplus \mathbf{h}_{\boldsymbol{b}} \oplus \mathbf{h}_{\boldsymbol{a}}$$

$$\mathbf{o} = MLP\ (\boldsymbol{h})$$

$$\mathbf{p} = softmax(\mathbf{o})$$

Contents

Global Structured Models

WestlakeNLP

- Neural CRF
  - CRF

$$P(T_{1:n}|W_{1:n}) = \frac{\exp\left(\vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n})\right)}{\sum_{T'_{1:n}} \exp\left(\vec{\theta} \cdot \vec{\phi}(T'_{1:n}, W_{1:n})\right)}$$

$$\vec{\theta} \cdot \vec{\phi}(T_{1:n}, W_{1:n}) = \sum_{i=1}^{n} \vec{\theta} \cdot \vec{\phi}(t_i, t_{i-1}, W_{1:n})$$

$$P(T_{1:n}|W_{1:n}) = \frac{\exp\left(\sum_{i=1}^{n} \vec{\theta} \cdot \vec{\phi}\,(t_i, t_{i-1}, W_{1:n})\right)}{\sum_{T'_{1:n}} \exp\left(\sum_{i=1}^{n} \vec{\theta} \cdot \vec{\phi}(t_{i'}, t_{i-1'}, W_{1:n})\right)}$$

  - Neural CRF – replace $\vec{\theta} \cdot \vec{\phi}$ with a neural score.

Global Structured Models

WestlakeNLP

- Neural CRF
  - A simple version

$$\mathbf{X}_{i-2:i+2} = emb(w_{i-2}) \oplus emb(w_{i-1}) \oplus emb(w_i) \oplus emb(w_{i+1}) \oplus emb(w_{i+2})$$

$$\mathbf{h}_i = f(W_{1:n}, i) = \tanh(\mathbf{W}^x \mathbf{X}_{i-2:i+2} + \mathbf{b}^x)$$

$$P(T_{1:n} \mid W_{1:n}) = \frac{\exp\left(\sum_{i=1}^{n}\left(\mathbf{U}(t_i)\mathbf{h}_i + b(t_i, t_{i-1})\right)\right)}{\sum_{T_1'^n}\left(\exp\left(\sum_{i=1}^{n}\left(\mathbf{U}(t_i')\mathbf{h}_i + b(t_i', t_{i-1}')\right)\right)\right)}$$

Global Structured Models

WestlakeNLP

- Neural CRF
  - Training $D = \{(W_i, T_i)\}|_{i=1}^{N}$

$$L(W_i, T_i, \Theta) = -\frac{1}{N}\sum_{i=1}^{N} \log P(T_i|W_i)$$

$$= -\frac{1}{N}\Big(\sum_{i=1}^{N}\Big(\sum_{j=1}^{|W_i|}\big(\mathbf{U}(t_j^i)\mathbf{h}_j^i + b(t_j^i, t_{j-1}^i)\big)\Big)\Big)$$

$$-\log\sum_{T'}\Big(\exp\Big(\sum_{j=1}^{|W_i|}\big(\mathbf{U}(t'_j)\mathbf{h}_j + b(t'_j, t'_{j-1})\big)\Big)\Big)$$

  - Use SGD

Global Structured Models

WestlakeNLP

- Neural CRF
  - A simple version
    - Training

$$\frac{\partial L(W_i, T_i, \Theta)}{\partial \mathrm{U}(l_k)} = -\Big(\sum_{j=1}^{|W_i|} \mathbf{h}_j^i \delta(t_j^i = l_k)$$

$$- \sum_{T'} \frac{\exp\Big(\sum_{j=1}^{|W_i|} \big(\mathrm{U}(t'_j)\mathbf{h}_j^i + b(t'_j, t'_{j-1})\big)\Big)}{\sum_{T''} \Big(\exp\Big(\sum_{j=1}^{|W_i|} \big(\mathrm{U}(t''_j)\mathbf{h}_j^i + b(t''_j, t''_{j-1})\big)\Big)\Big)} \sum_{j=1}^{|W_i|} \mathbf{h}_j \delta(t'_j = l_k)\Big)$$

$$= -\Big(\sum_{j=1}^{|W_i|} \mathbf{h}_j^i \delta(t_j^i = l_k) - \sum_{T'} P(T' \mid w_i)\mathbf{h}_j^i \delta(t'_j = l_k)\Big)$$

$$= -\sum_{j=1}^{|W_i|} \Big(\mathbf{h}_j^i \delta(t_j^i = l_k) - \sum_{T'} \big(P(T' \mid w_i)\mathbf{h}_j^i \delta(t'_j = l_k)\big)\Big)$$

$$= -\sum_{j=1}^{|W_i|} \Big(\mathbf{h}_j^i \delta(t_j^i = l_k) - \mathbb{E}_{T' \sim P(T'|W_i)} \mathbf{h}_j^i \delta(t'_j = l_k)\Big)$$

$$\mathbb{E}_{T' \sim P(T'|W_i)} \mathbf{h}_j^i \delta\big(t'_j = l_k\big) = \mathbb{E}_{t'_j \sim P(t'_j|W_i)} \mathbf{h}_j^i \delta\big(t'_j = l_k\big)$$

Global Structured Models

- Neural CRF
  - A simple version
    - Training

$$\frac{\partial L(W_i, T_i, \Theta)}{\partial \mathbf{h}^i_j} = \mathrm{U}(t^i_j) - \frac{\exp\left(\sum_{j=1}^{|W_i|}\left(\mathrm{U}(t^i_j)\mathbf{h}^i_j + b(t^i_j, t^i_{j-1})\right)\right)}{\sum_{T'}\left(\exp\left(\sum_{j=1}^{|W_i|}\left(\mathrm{U}(t'_j)\mathbf{h}^i_j + b(t'_j, t'_{j-1})\right)\right)\right)}\mathrm{U}(t'_j)$$

$$= \mathrm{U}(t^i_j) - \sum_{T'} P(T' \mid W_i)\mathrm{U}(t'_j)$$

$$= \mathrm{U}(t^i_j) - \mathbb{E}_{T' \sim P(T'|W_i)}\mathrm{U}(t'_j)$$

$$= \mathrm{U}(t^i_j) - \mathbb{E}_{t'_j \sim P(t'_j|W_i)}\mathrm{U}(t'_j)$$

Global Structured Models

WestlakeNLP

- Neural CRF
  - A more complex version BiLSTM CRF.

$$\boldsymbol{H}_{1:n} = BiLSTM(\boldsymbol{X}_{1:n})$$

$$P(T_{1:n} \mid W_{1:n}) = \frac{\exp\Big(\sum_{i=1}^{n}\Big(\mathrm{U}(t_i)\mathbf{h}_i + b(t_i, t_{i-1})\Big)\Big)}{\sum_{T'}\Big(\exp\Big(\sum_{i=1}^{n}\Big(\mathrm{U}(t'_i)\mathbf{h}_i + b(t'_i, t'_{i-1})\Big)\Big)\Big)}$$

(add a sequence encoder)

  - Gradient calculation remains similar.

Global Structured Models

WestlakeNLP

- Neural CRF
  - More variants?
  - Consider label embedding

Global Structured Models

- Neural CRF
  - Tree CRF

$$P(T|W) = \frac{\exp\left(\vec{\theta} \cdot \vec{\phi}(W, T)\right)}{\sum_{T' \in Gen(W)} \exp\left(\vec{\theta} \cdot \vec{\phi}(W, T')\right)}$$

$$\vec{\phi}(W,T) = \sum_{r \in T} \vec{\phi}\,(W,r)$$

$$P(T \mid W) = \frac{\exp\left(\sum_{r \in T} \vec{\theta} \cdot \vec{\phi}(W,r)\right)}{\sum_{T' \in Gen(W)} \exp\left(\sum_{r' \in T'} \vec{\theta} \cdot \vec{\phi}(W,r')\right)}$$

  - Neural version – replace $\vec{\theta} \cdot \vec{\phi}$ with a neural score.

Global Structured Models

WestlakeNLP

- Neural CRF
  - Neural version using label embedding parameterization

$$P(T|W) = \frac{\exp\left(\sum_{r \in T} f(W, r)\right)}{\sum_{T' \in Gen(W)} \exp\left(\sum_{r' \in T'} f(W, r')\right)}$$

$$f(W, r) = f\left(W, c, \rightarrow, c_1, c_2, bb'e\vec{\theta}\right) = \vec{\tau}(W, b, b', e)^T \mathbf{W}^f \vec{\gamma}(c \rightarrow c_1 c_2)$$

$$\vec{\tau}(W, b, b', e) = ReLU\left(\mathbf{W}^w(\mathbf{h}_b \oplus \mathbf{h}_{b'-1} \oplus \mathbf{h}_e)\right)$$

$$\vec{\gamma}(c \rightarrow c_1 c_2)$$
$$= ReLU\left(\mathbf{W}^a emb^s(c) + \mathbf{W}^l emb^s(c_1) + \mathbf{W}^r emb^s(c_2) + \mathbf{b}\right)$$

Global Structured Models

WestlakeNLP

- Neural CRF
  - Training $D = \{(W_i, T_i)\}|_{i=1}^{N}$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log P(T_i \mid W_i) = -\frac{1}{N}\sum_{i=1}^{N}\log \frac{\exp\left(\sum_{r\in T_i}\vec{\tau}(r)^T \mathbf{W}^f \vec{\gamma}(r)\right)}{\sum_{T_i'\in Gen(W_i)}\exp\left(\sum_{r'\in T_i'}\vec{\tau}(r')^T \mathbf{W}^f \vec{\gamma}(r')\right)}$$

  - Use SGD

Global Structured Models

WestlakeNLP

- Neural CRF
  - Tree CRF
    - Training

$$\frac{\partial L}{\partial \mathbf{W}^f} = -\Big(\sum_{r \in T_i} \vec{\tau}(r)\vec{\gamma}(r)^T - \sum_{T'_i \in Gen(W_i)} P(T'_i | W_i) \sum_{r' \in T'_i} \vec{\tau}(r')\vec{\gamma}(r')^T\Big)$$

$$= -\left(\sum_{r \in T_i} \vec{\tau}(r)\vec{\gamma}(r)^T - \mathbb{E}_{T'_i \sim P(T'_i|W_i)} \sum_{r' \in T'_i} \vec{\tau}(r')\vec{\gamma}(r')^T\right)$$

$$\frac{\partial L}{\partial \vec{\tau}(r)} = -\left(\sum_{r \in T_i} \mathbf{W}^f \vec{\gamma}(r) - \sum_{T'_i \in Gen(W_i)} P(T'_i|W_i) \sum_{r' \in T'_i} \mathbf{W}^f \vec{\gamma}(r')\right)$$

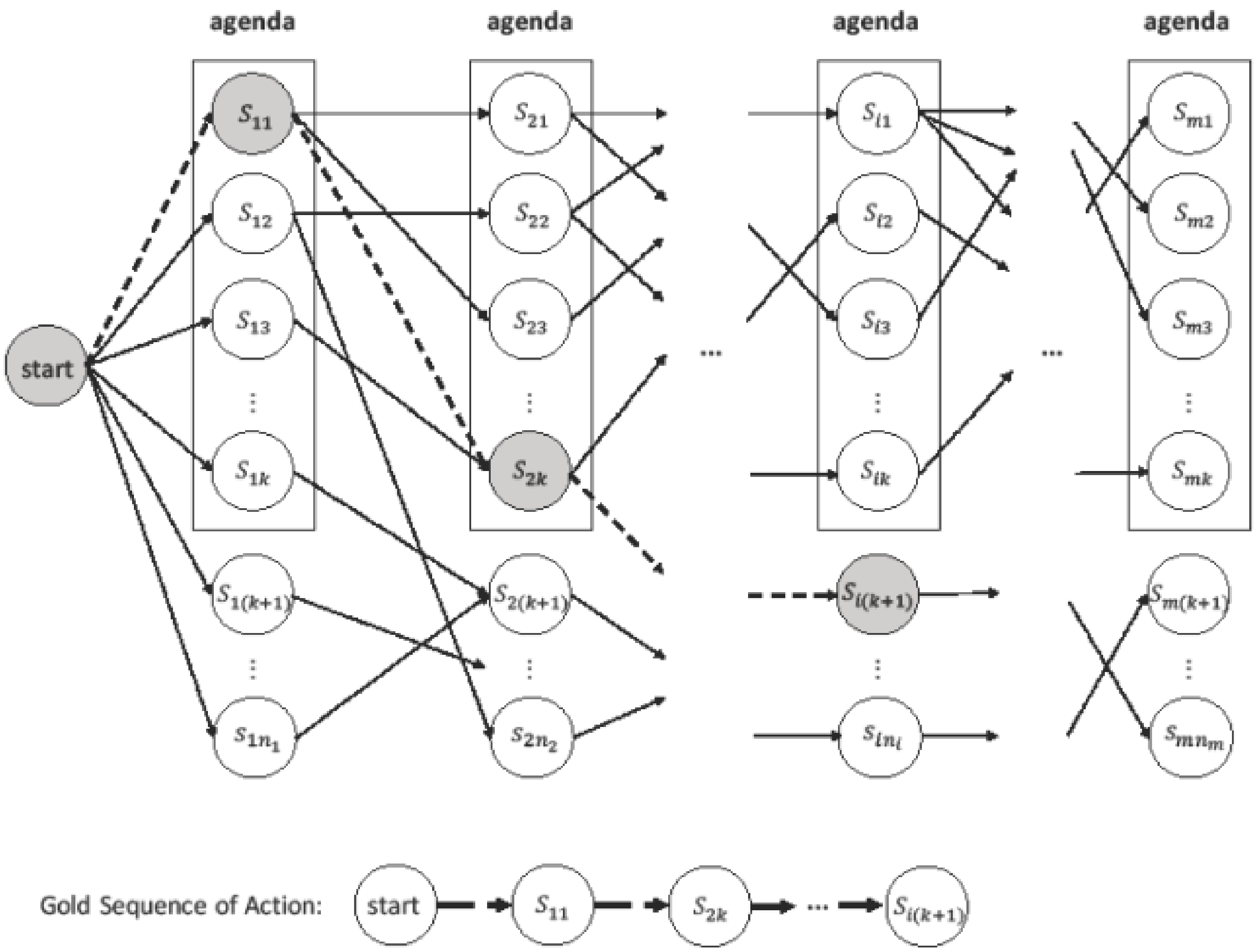
$$= -\left(\sum_{r \in T_i} \mathbf{W}^f \vec{\gamma}(r) - \mathbb{E}_{T'_i \sim P(T'_i|W_i)} \sum_{r' \in T'_i} \mathbf{W}^f \vec{\gamma}(r')\right)$$

Contents

WestlakeNLP

Global Structured Models

- Neural Transition-Based Models with Global Normalization



Global Structured Models

- Neural Transition-Based Models with Global Normalization
- Linear version

$$score(A_{1:i}) = \sum_{j=1}^{i} score(a_j) = \sum_{j=1}^{i} \vec{\theta} \cdot \vec{\phi}(s_{j-1} \cdot a_j)$$

- Neural version

$$score(A_{1:i}) = \sum_{j=1}^{i} score(a_j) = \sum_{j=1}^{i} \mathbf{o}_j [a_j]$$

$$A_{1:i} = a_1, a_2, \ldots, a_i$$

Global Structured Models

- Neural Transition-Based Models with Global Normalization
  - Training

$$P(A_{1:i}|W_{1:n}) = softmax(A_{1:i}, A'_{1:i} \in gen(W_{1:n}, i)) = \frac{\exp\left(\sum_{j=1}^{i} \mathbf{o}_j \, [a_j]\right)}{\sum_{A'_{1:i}} \exp\left(\sum_{j=1}^{i} \mathbf{o'}_j \, [a_j]\right)}$$

$$L = -\log P(A_{1:i}|W_{1:n})$$

$$= -\log \frac{\exp\left(\sum_{j=1}^{i} \mathbf{o}_j \, [a_j]\right)}{\sum_{A'_{1:i}} \exp\left(\sum_{j=1}^{i} \mathbf{o}_{j'} \, [a_j]\right)}$$

$$= -\log \frac{\exp\left(\sum_{j=1}^{i} \mathbf{o}_j \, [a_j]\right)}{Z}$$

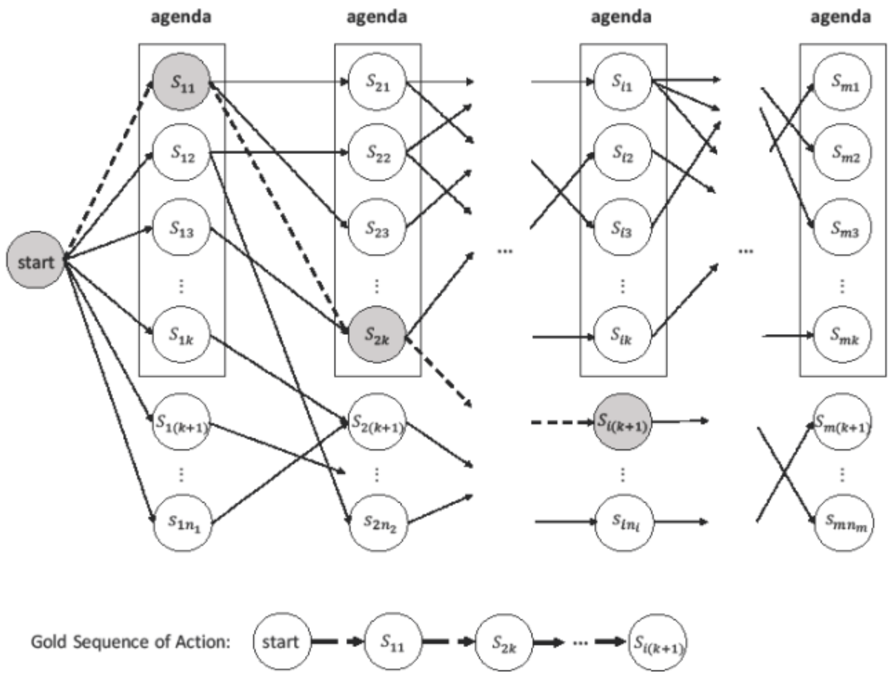
$$= \log Z - \sum_{j=1}^{i} \mathbf{o}_j \, [a_j]$$

where $Z = \sum_{A'_{1:i}} \exp\left(\sum_{j=1}^{i} \mathbf{o}_j \, [a'_j]\right)$

Global Structured Models

- Neural Transition-Based Models with Global Normalization
  - $A'_{1:i}$ is exponential to i
  - Contrastive estimation

$$Z'(x_i,\theta) = \sum_{A'_{1:i} \in [s_{i,1},\ldots,s_{i,n_i}]} \exp\left(\sum_{j=1}^{i} \mathbf{o}_j\,[a'_j]\right)$$



Summary

- Local graph-based neural models
- Local transition-based neural models
- Global graph-based neural models
- Global transition-based neural models