Language Technologies Institute

Carnegie Mellon University

Algorithms for NLP

CS 11-711 · Fall 2020

Lecture 9: CRFs, neural sequence labeling

Emma Strubell

Announcements

- **Project 2 released today after class**: sequence labeling.
  - Due: October 16.
  - You will implement part-of-speech taggers for English and Norwegian:
  - HMM, BiLSTM, and BiLSTM-CRF.
- Friday's recitation will be an overview of P2.

Recap

Recap

- **HMMs**: Natural extension of Naive Bayes to sequence labeling

Recap

- **HMMs**: Natural extension of Naive Bayes to sequence labeling
  - Hard to add rich features of the input, e.g. affixes, capitalization, …

Recap

- **HMMs**: Natural extension of Naive Bayes to sequence labeling
  - Hard to add rich features of the input, e.g. affixes, capitalization, …
- Would like to train a **discriminative model**, like logistic regression, to directly model the conditional probability of labels given inputs.

Recap

- **HMMs**: Natural extension of Naive Bayes to sequence labeling
  - Hard to add rich features of the input, e.g. affixes, capitalization, …
- Would like to train a **discriminative model**, like logistic regression, to directly model the conditional probability of labels given inputs.
  - Logistic regression (MEMMs) suffer from the **label bias problem**.

Recap

- **HMMs**: Natural extension of Naive Bayes to sequence labeling
  - Hard to add rich features of the input, e.g. affixes, capitalization, …
- Would like to train a **discriminative model**, like logistic regression, to directly model the conditional probability of labels given inputs.
  - Logistic regression (MEMMs) suffer from the **label bias problem**.
- Solution: **linear-chain CRFs**.

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m)$$

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m)$$

decompose into local scores

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m)$$

decompose into local scores

$$= \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

Conditional random fields (CRFs)

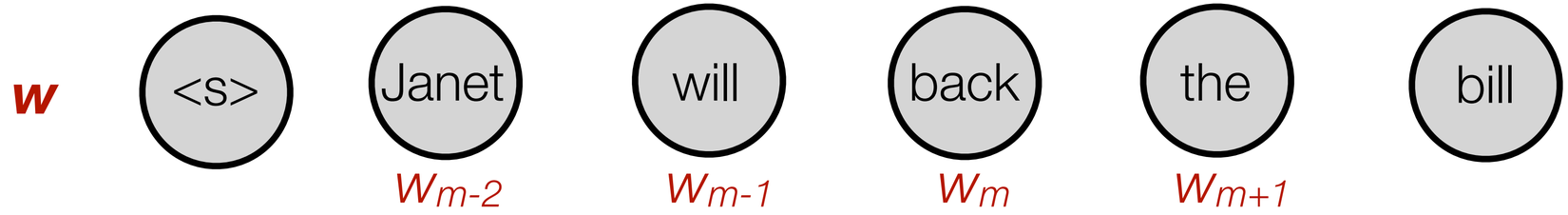
- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

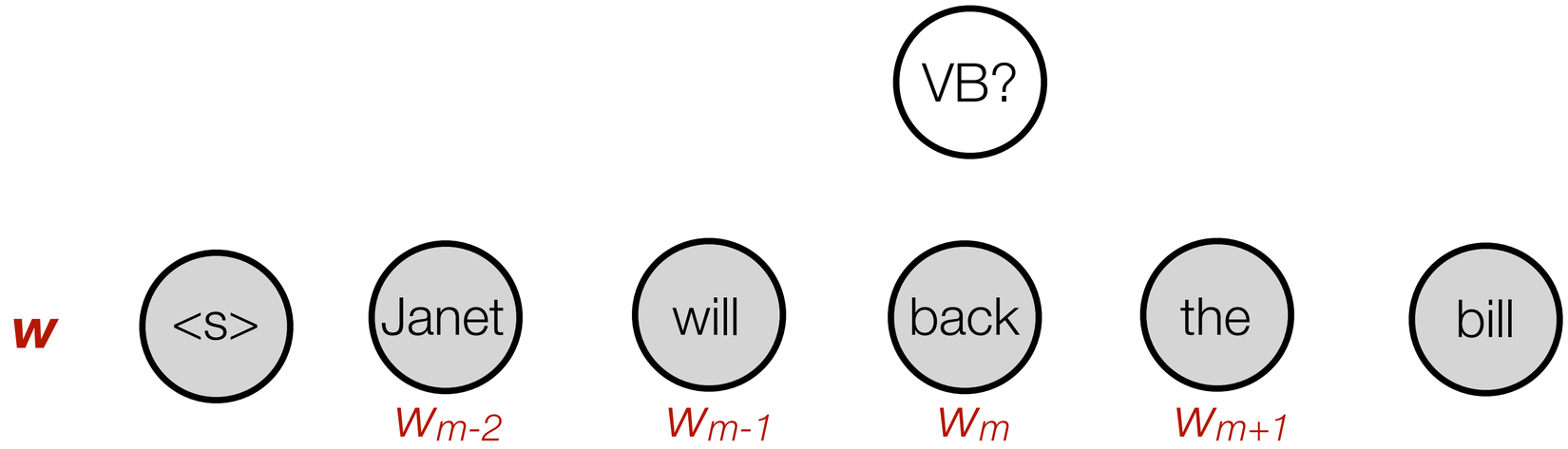
$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$



Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

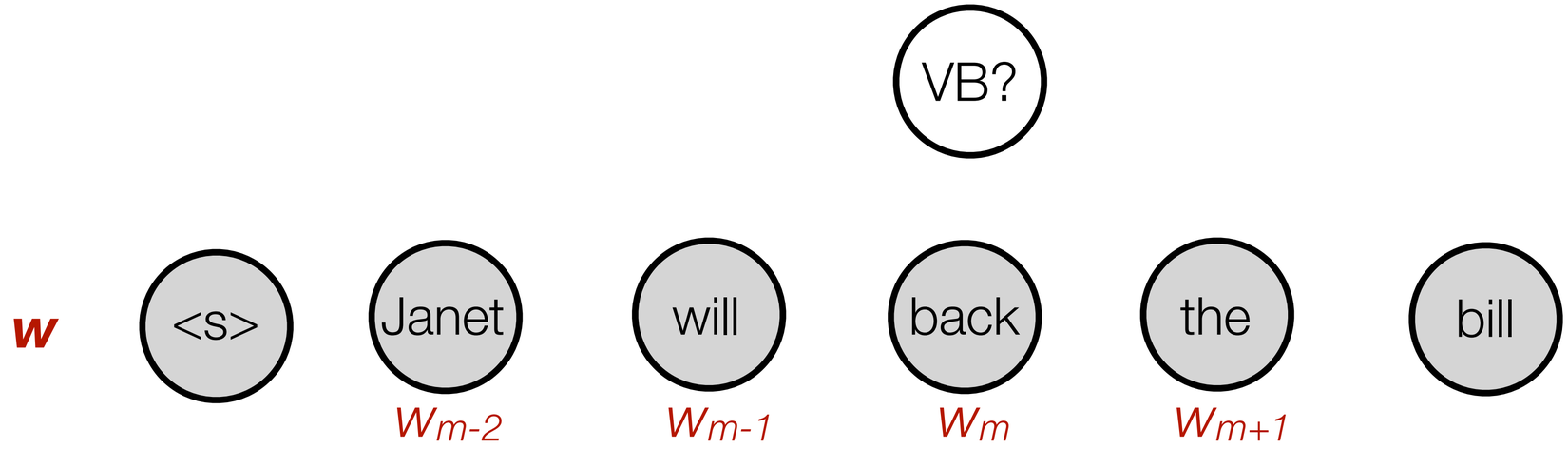


Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0



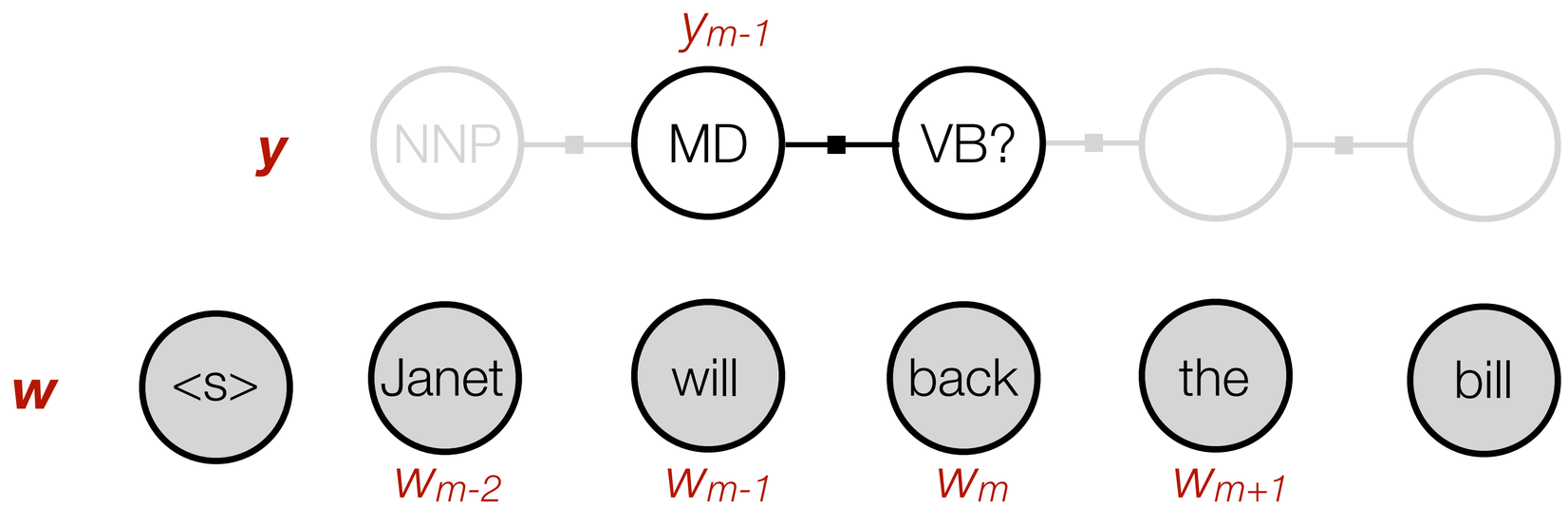
Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

f(***w***, VB, MD, 3)

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0



Conditional random fields (CRFs)

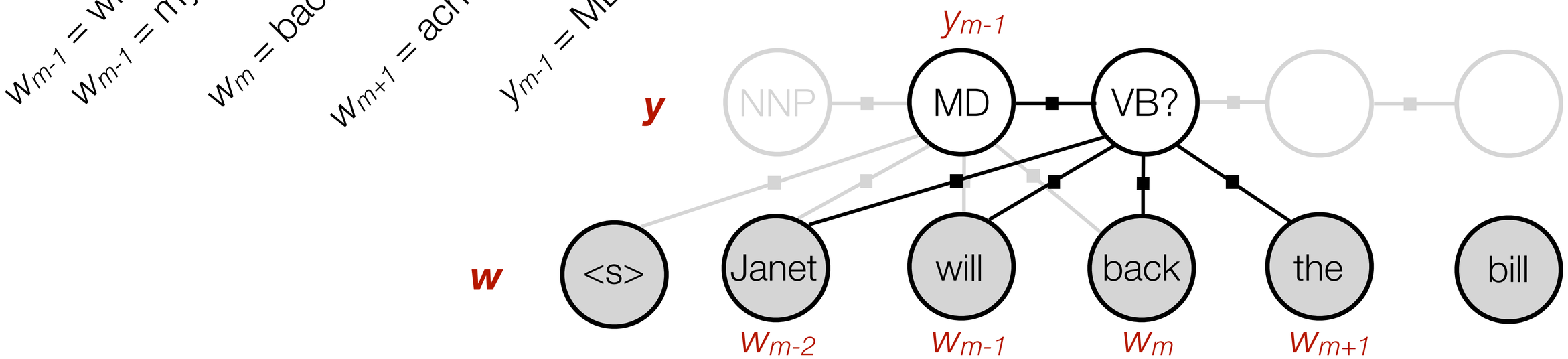
- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

$\boldsymbol{f}(\boldsymbol{w}$, VB, MD, 3)
$= \boldsymbol{f}(w_1, w_2, w_3, w_4$, VB, MD)

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0

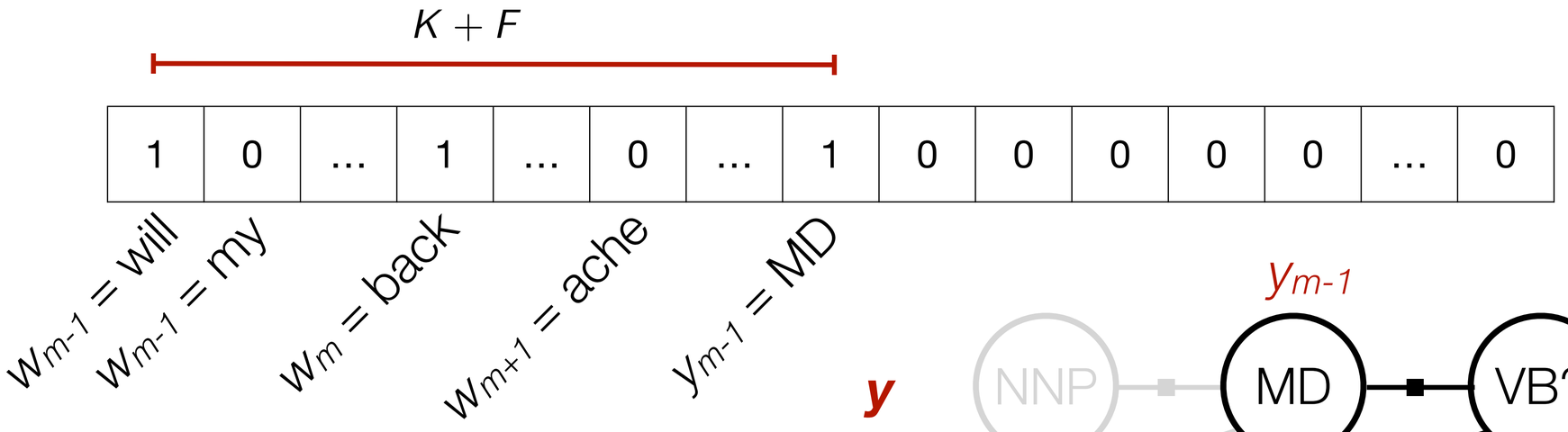
w_{m-1} = will
w_{m-1} = my
w_m = back
w_{m+1} = ache
y_{m-1} = MD



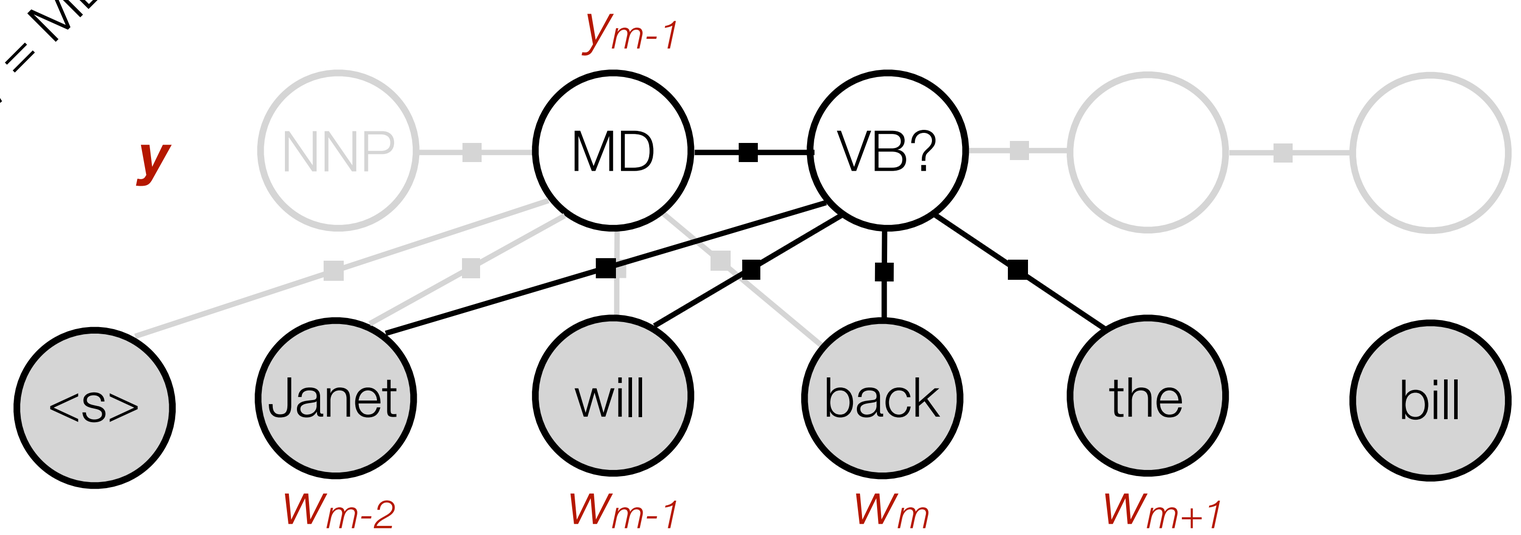
Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$



f(***w***, VB, MD, 3)
= ***f***(w_1, w_2, w_3, w_4, VB, MD)



Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

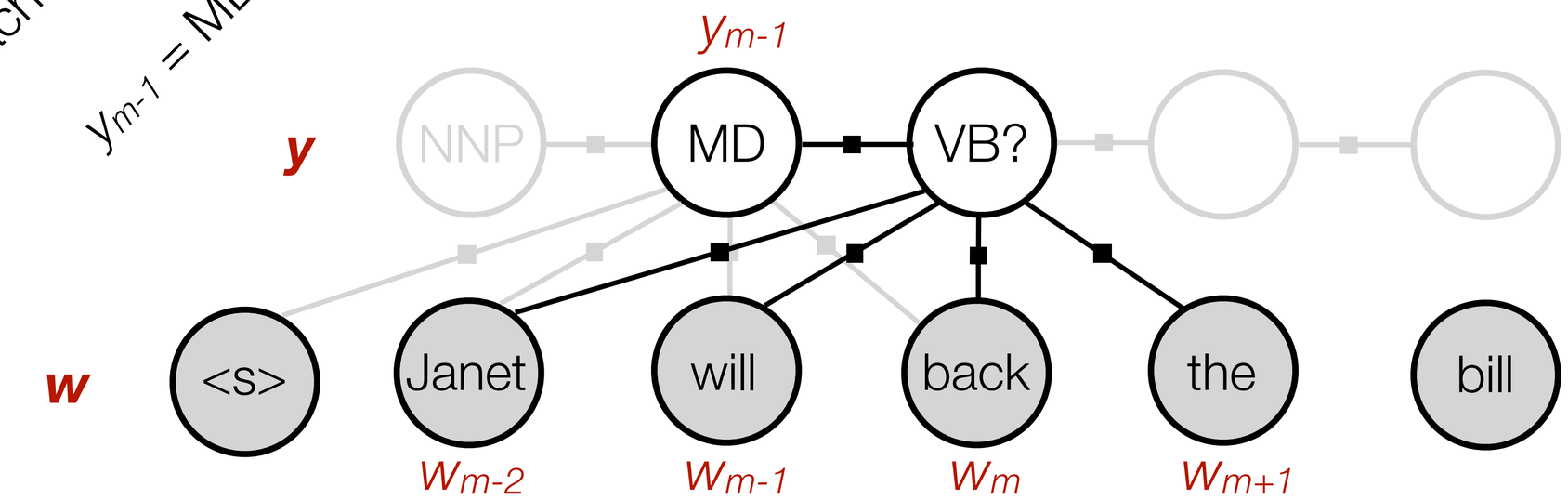
labels

$K + F$

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0

w_{m-1} = will
w_{m-1} = my
w_m = back
w_{m+1} = ache
y_{m-1} = MD

$\boldsymbol{f}(\boldsymbol{w}$, VB, MD, 3)
$= \boldsymbol{f}(w_1, w_2, w_3, w_4$, VB, MD)



Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

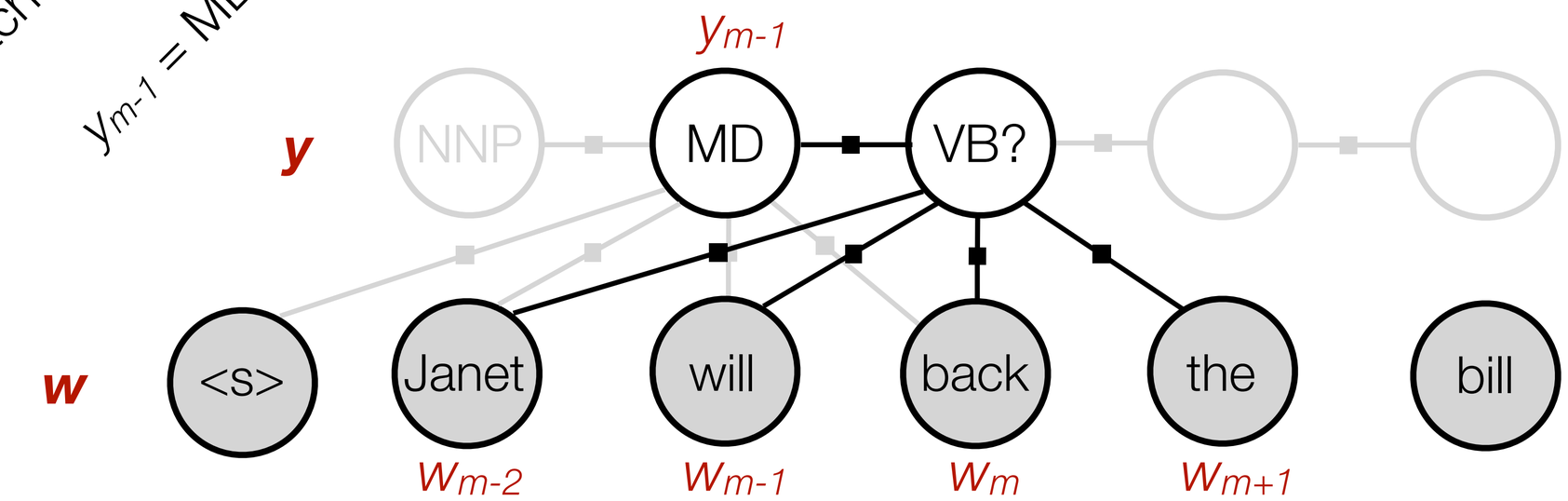
labels # features

$K + F$

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0

w_{m-1} = will
w_{m-1} = my
w_m = back
w_{m+1} = ache
y_{m-1} = MD

$\boldsymbol{f}(\boldsymbol{w}, \text{VB}, \text{MD}, 3)$
$= \boldsymbol{f}(w_1, w_2, w_3, w_4, \text{VB}, \text{MD})$



Conditional random fields (CRFs)

- **Linear-chain CRFs**: Globally-normalized discriminative sequence labeling models!

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

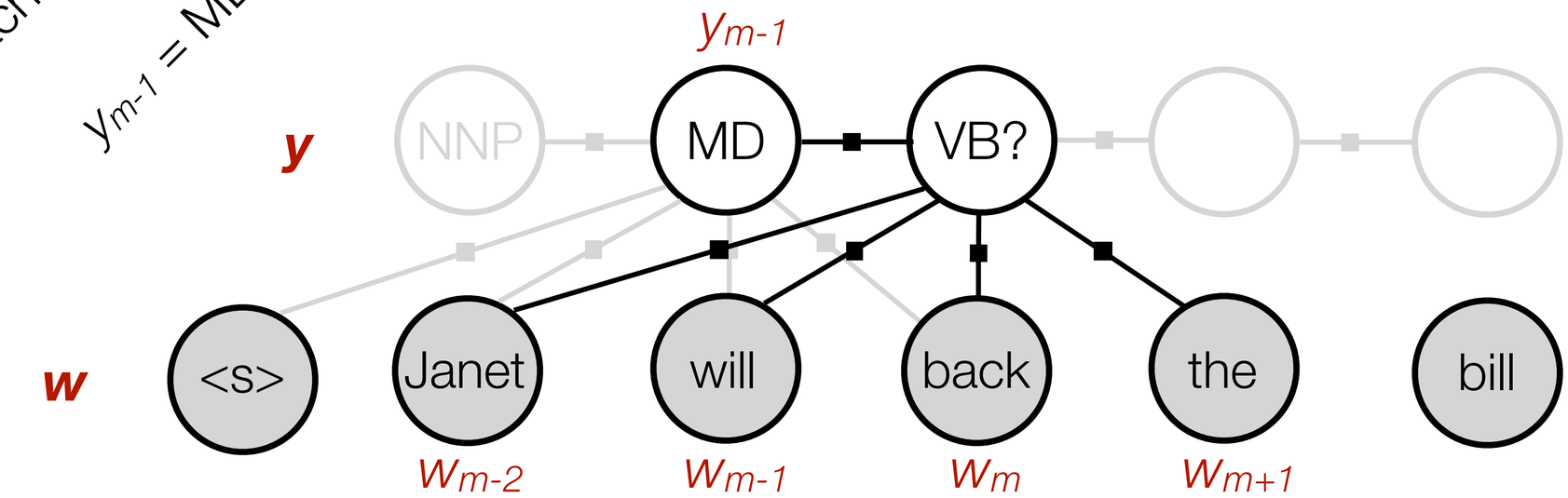
labels # features

$K + F$ $(K-1)(K+F)$

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0

w_{m-1} = will
w_{m-1} = my
w_m = back
w_{m+1} = ache
y_{m-1} = MD

$\boldsymbol{f}(\boldsymbol{w}$, VB, MD, 3)
$= \boldsymbol{f}(w_1, w_2, w_3, w_4$, VB, MD)



Conditional random fields (CRFs)

- **Decoding:**

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log P(\mathbf{y} \mid \mathbf{w})$$

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

$$\begin{aligned}\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y} \mid \mathbf{w}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp \Psi(\mathbf{y}', \mathbf{w})\end{aligned}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

$$\begin{aligned}\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y} \mid \mathbf{w}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp \Psi(\mathbf{y}', \mathbf{w})\end{aligned}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

$$\begin{aligned}
\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y} \mid \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp \Psi(\mathbf{y}', \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w})
\end{aligned}$$

same for all settings of ***y***

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

$$\begin{aligned}
\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y} \mid \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp \Psi(\mathbf{y}', \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) \qquad \text{same for all settings of } \boldsymbol{y} \\
&= \operatorname*{argmax}_{\mathbf{y}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m)
\end{aligned}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

$$\begin{aligned}
\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y} \mid \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) - \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp \Psi(\mathbf{y}', \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{y}, \mathbf{w}) \qquad \text{same for all settings of } \boldsymbol{y} \\
&= \operatorname*{argmax}_{\mathbf{y}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\end{aligned}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{m=1}^{M+1} \boxed{\psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})}$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{m=1}^{M+1} \boxed{\psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})}$$

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})$$

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \sum_{m=1}^{M+1} \boxed{\psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})}$$

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

$$= \max_{\mathbf{y}_M} s_{M+1}(\langle /s \rangle, y_M) + \max_{\mathbf{y}_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})$$

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

$$= \max_{\mathbf{y}_M} s_{M+1}(\langle /s \rangle, y_M) + \max_{\mathbf{y}_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

score of best tag sequence of length M-1

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})$$

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

$$= \max_{\mathbf{y}_M} s_{M+1}(\langle /s \rangle, y_M) + \max_{\mathbf{y}_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

score of most probable extension y_M

score of best tag sequence of length M-1

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{m=1}^{M+1} \boxed{\psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})}$$

same subproblem!

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

$$= \max_{\mathbf{y}_M} s_{M+1}(\langle /s \rangle, y_M) + \max_{\mathbf{y}_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

score of most probable extension y_M

score of best tag sequence of length M-1

Conditional random fields (CRFs)

- **Decoding:** Direct application of Viterbi!

ψ includes transition and emission/unary scores

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{m=1}^{M+1} \psi(\mathbf{w}, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})$$

same subproblem!

score of best tag sequence of length M

$$\max_{\mathbf{y}_{1:M}} \Psi(\mathbf{w}, \mathbf{y}_{1:M}) = \max_{\mathbf{y}_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

$$= \max_{\mathbf{y}_M} s_{M+1}(\langle /s \rangle, y_M) + \max_{\mathbf{y}_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

score of best tag sequence of length M-1

score of most probable extension y_M

Learning in CRFs

Learning in CRFs

- As with logistic regression, weights θ are learned by minimizing negative log likelihood:

$$\ell = -\sum_{i=1}^{N} \log P(\mathbf{y}^{(i)} \mid \mathbf{w}^{(i)}; \theta)$$

Learning in CRFs

- As with logistic regression, weights θ are learned by minimizing negative log likelihood:

$$\ell = -\sum_{i=1}^{N} \log P(\mathbf{y}^{(i)} \mid \mathbf{w}^{(i)}; \theta)$$

$$= -\sum_{i=1}^{N} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}^{(i)}) + \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w}^{(i)})} \exp\left(\boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}')\right)$$

Learning in CRFs

- As with logistic regression, weights θ are learned by minimizing negative log likelihood:

$$\ell = -\sum_{i=1}^{N} \log P(\mathbf{y}^{(i)} \mid \mathbf{w}^{(i)}; \theta)$$

$$= -\sum_{i=1}^{N} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}^{(i)}) + \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w}^{(i)})} \exp\left(\boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}')\right)$$

sum over all possible labelings

Learning in CRFs

- As with logistic regression, weights θ are learned by minimizing negative log likelihood:

$$\ell = -\sum_{i=1}^{N} \log P(\mathbf{y}^{(i)} \mid \mathbf{w}^{(i)}; \theta)$$

$$= -\sum_{i=1}^{N} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}^{(i)}) + \log \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w}^{(i)})} \exp\left(\boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}^{(i)}, \mathbf{y}')\right)$$

sum over all possible labelings

- Can be computed efficiently using **forward algorithm**.

Learning in CRFs

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**.

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1})$$

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

- As in decoding / Viterbi, can be decomposed into recursive substructure:

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{\mathbf{y}_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})$$

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

- As in decoding / Viterbi, can be decomposed into recursive substructure:

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{\mathbf{y}_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})$$

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})$$

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

- As in decoding / Viterbi, can be decomposed into recursive substructure:

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{\mathbf{y}_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})$$

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})$$

sum instead of max

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

- As in decoding / Viterbi, can be decomposed into recursive substructure:

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{\mathbf{y}_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})$$

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})$$

sum instead of max

Viterbi is a special case of the *max-product algorithm*, forward is a special case of the *sum-product algorithm*.

Learning in CRFs

- Likelihood can be computed efficiently using **forward algorithm**. Define:

$$\alpha_m(y_m) = \sum_{\mathbf{y}_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{\mathbf{y}_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})$$

- As in decoding / Viterbi, can be decomposed into recursive substructure:

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{\mathbf{y}_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})$$

$$= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})$$

sum instead of max

Viterbi is a special case of the *max-product algorithm*, forward is a special case of the *sum-product algorithm*.

$$v_m(k) = \bigoplus_{k' \in \mathcal{Y}} s_m(k, k') \otimes v_{m-1}(k')$$

Learning in CRFs

Learning in CRFs

- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

Learning in CRFs

- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

Learning in CRFs

- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

Learning in CRFs

- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

- In the Olden Days, would use the **forward-backward algorithm** to compute expected counts.

Learning in CRFs

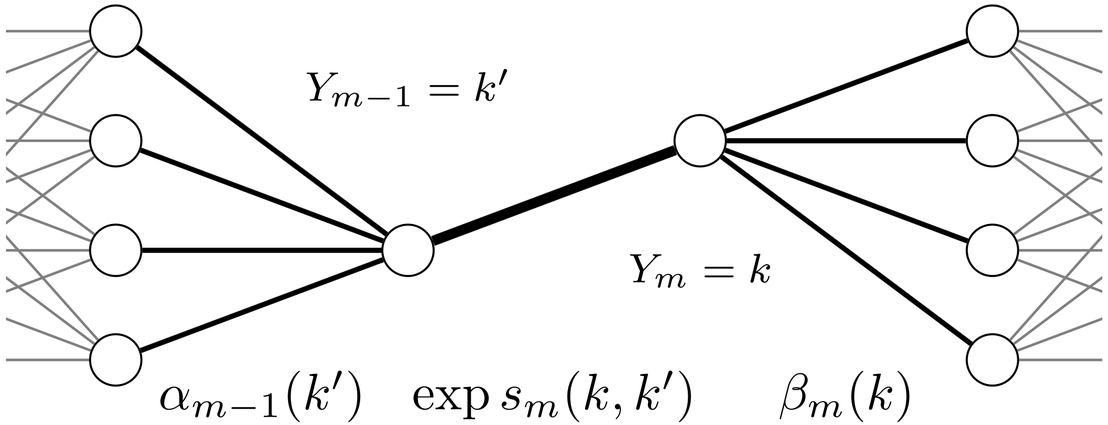
- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

- In the Olden Days, would use the **forward-backward algorithm** to compute expected counts.



Learning in CRFs

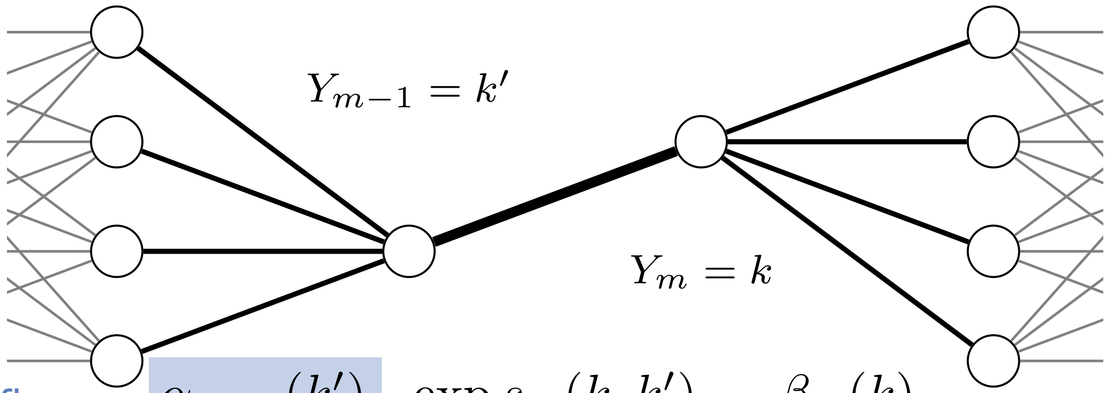
- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

- In the Olden Days, would use the **forward-backward algorithm** to compute expected counts.



forward score:
sum over all prefixes

Learning in CRFs

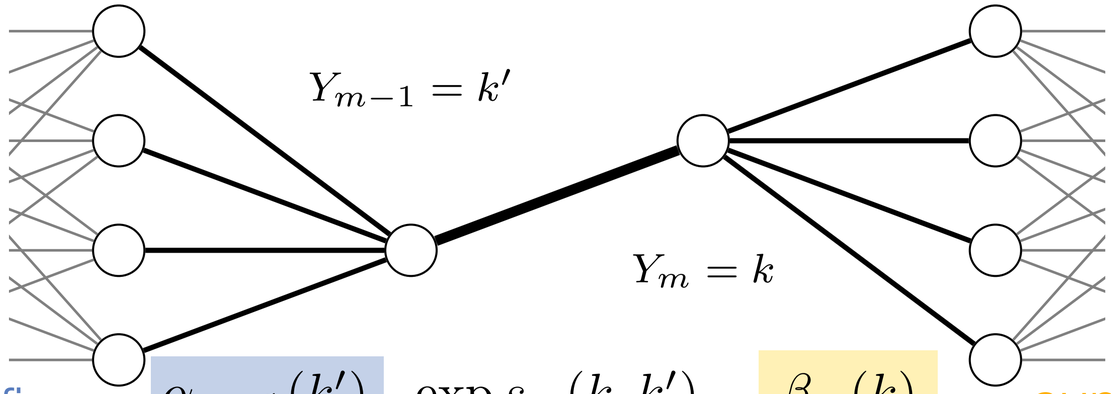
- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

- In the Olden Days, would use the **forward-backward algorithm** to compute expected counts.



forward score: sum over all prefixes

$\alpha_{m-1}(k')$ $\exp s_m(k, k')$ $\beta_m(k)$

backward score: sum over all suffixes

Learning in CRFs

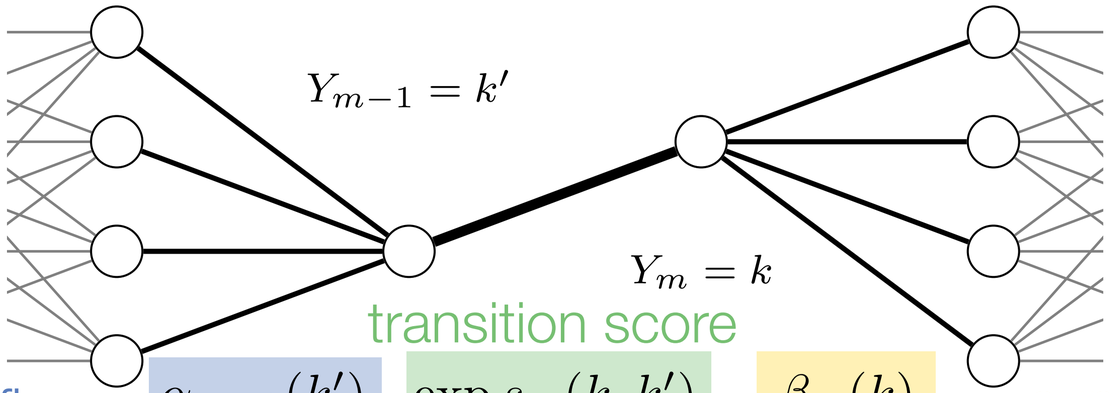
- As in logistic regression, gradient of the likelihood w.r.t. parameters is difference between observed and expected feature counts:

$$\frac{\delta \ell}{\delta \theta_j} = \sum_{i=1}^{N} E[f_j(\mathbf{w}^{(i)}, \mathbf{y})] - f_j(\mathbf{w}^{(i)}, \mathbf{y}^{(i)})$$

count of feature *j* for token sequence $\boldsymbol{w}^{(i)}$, tag sequence $\boldsymbol{y}^{(i)}$

- Gradients can be computed by automatic differentiation!

- In the Olden Days, would use the **forward-backward algorithm** to compute expected counts.



forward score: sum over all prefixes

$\alpha_{m-1}(k')$

transition score

$\exp s_m(k, k')$

$\beta_m(k)$

backward score: sum over all suffixes

Better features for sequence labeling?

- Until now: hand-engineered features:

Better features for sequence labeling?

- Until now: hand-engineered features:

Lexical	$f_{i\pm\{0,1,2,3\}}$, $(m_{i-2,i-1})$, $(m_{i-1,i})$, $(m_{i-1,i+1})$, $(m_{i,i+1})$, $(m_{i+1,i+2})$, $(m_{i-2,i-1,i})$, $(m_{i-1,i,i+1})$, $(m_{i,i+1,i+2})$, $(m_{i-2,i-1,i+1})$, $(m_{i-1,i+1,i+2})$
POS	$p_{i-\{3,2,1\}}$, $a_{i+\{0,1,2,3\}}$, $(p_{i-2,i-1})$, $(a_{i+1,i+2})$, (p_{i-1}, a_{i+1}), (p_{i-2}, p_{i-1}, a_i), $(p_{i-2}, p_{i-1}, a_{i+1})$, (p_{i-1}, a_i, a_{i+1}), $(p_{i-1}, a_{i+1}, a_{i+2})$
Affix	$c_{:1}$, $c_{:2}$, $c_{:3}$, $c_{n:}$, $c_{n-1:}$, $c_{n-2:}$, $c_{n-3:}$
Binary	initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

Better features for sequence labeling?

- Until now: hand-engineered features:

Lexical	$f_{i\pm\{0,1,2,3\}}$, $(m_{i-2,i-1})$, $(m_{i-1,i})$, $(m_{i-1,i+1})$, $(m_{i,i+1})$, $(m_{i+1,i+2})$, $(m_{i-2,i-1,i})$, $(m_{i-1,i,i+1})$, $(m_{i,i+1,i+2})$, $(m_{i-2,i-1,i+1})$, $(m_{i-1,i+1,i+2})$
POS	$p_{i-\{3,2,1\}}$, $a_{i+\{0,1,2,3\}}$, $(p_{i-2,i-1})$, $(a_{i+1,i+2})$, (p_{i-1}, a_{i+1}), (p_{i-2}, p_{i-1}, a_i), $(p_{i-2}, p_{i-1}, a_{i+1})$, (p_{i-1}, a_i, a_{i+1}), $(p_{i-1}, a_{i+1}, a_{i+2})$
Affix	$c_{:1}$, $c_{:2}$, $c_{:3}$, $c_{n:}$, $c_{n-1:}$, $c_{n-2:}$, $c_{n-3:}$
Binary	initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

$$\mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m) =$$

1	0	…	1	…	0	…	1	0	0	0	0	0	…	0

w_{m-1} = will
w_{m-1} = my
w_m = back
w_{m+1} = ache
y_{m-1} = MD

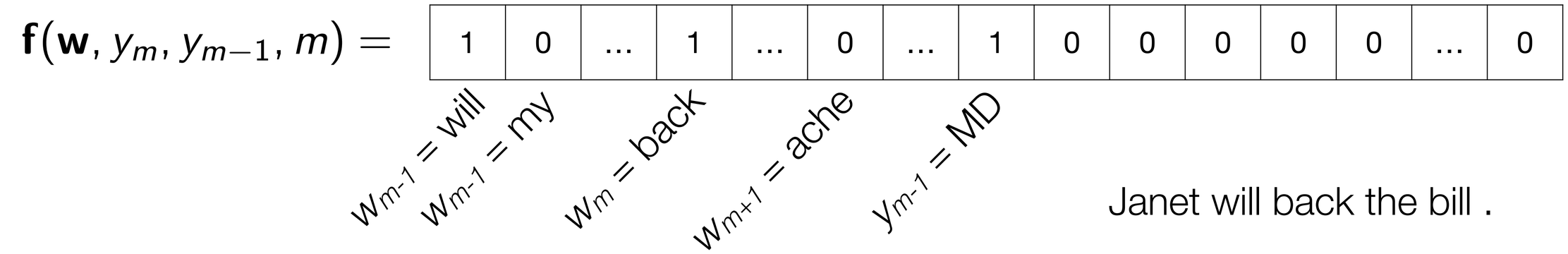
Janet will back the bill .

Better features for sequence labeling?

- Until now: hand-engineered features:

pros:

cons:



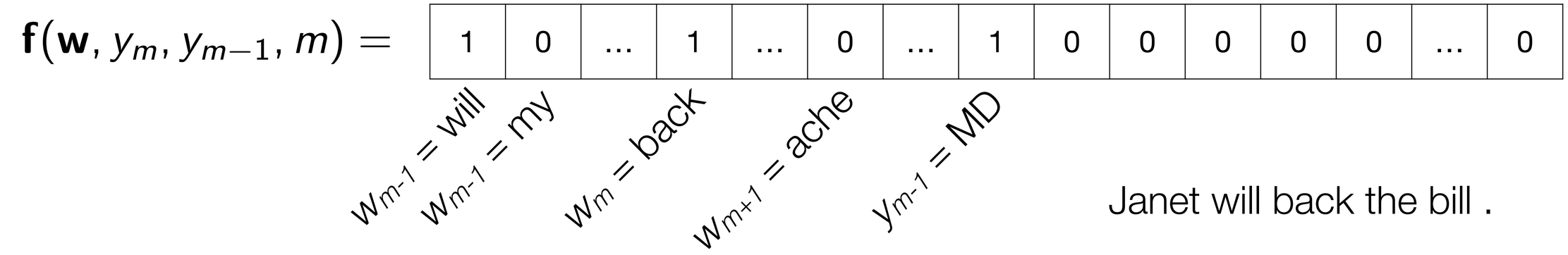
Better features for sequence labeling?

- Until now: hand-engineered features:

pros:

- interpretable, explainable
- can generalize well
- fast training and inference
- channel domain knowledge

cons:



Better features for sequence labeling?

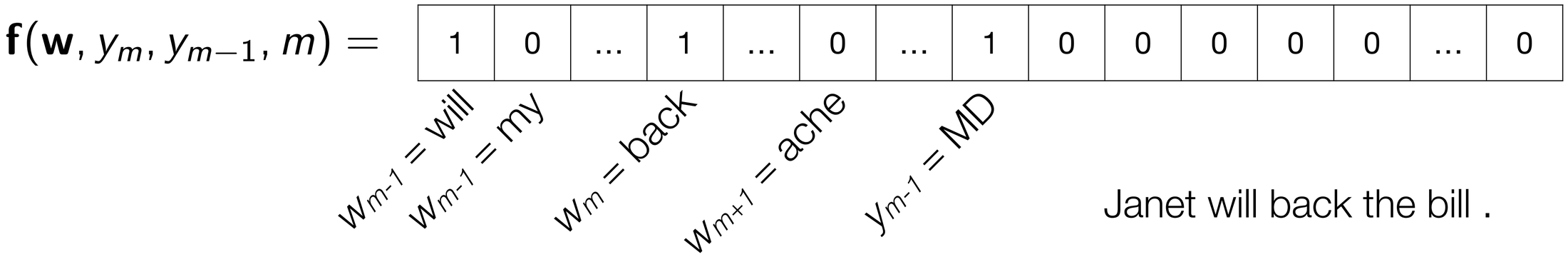
- Until now: hand-engineered features:

pros:

- interpretable, explainable
- can generalize well
- fast training and inference
- channel domain knowledge

cons:

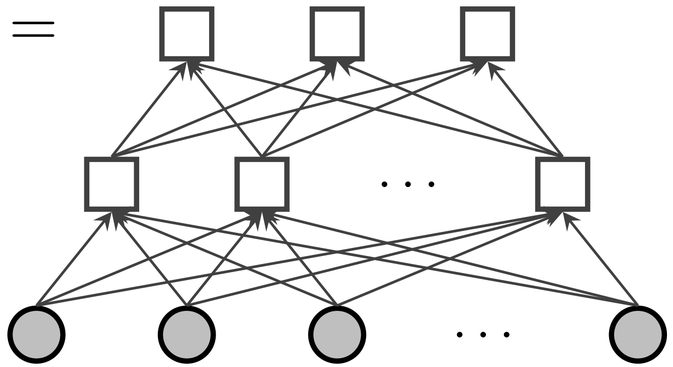
- can be sparse/high variance
- lack of shared representations
- task-specific
- worse performance



Neural sequence labeling

- Parameterize ***f*** with a (deep) neural network.

$$\mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m) =$$



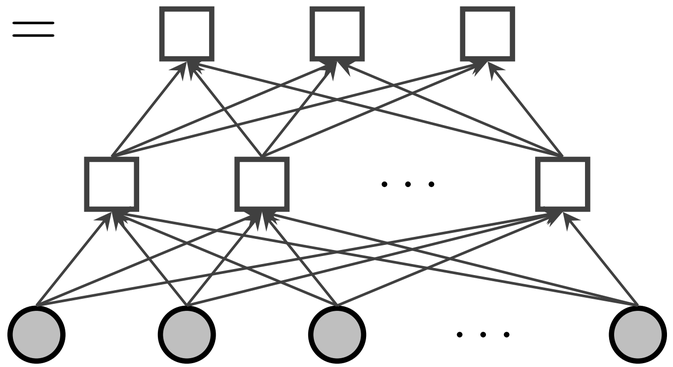
Neural sequence labeling

- Parameterize ***f*** with a (deep) neural network.

pros:

cons:

$$\mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m) =$$



Neural sequence labeling

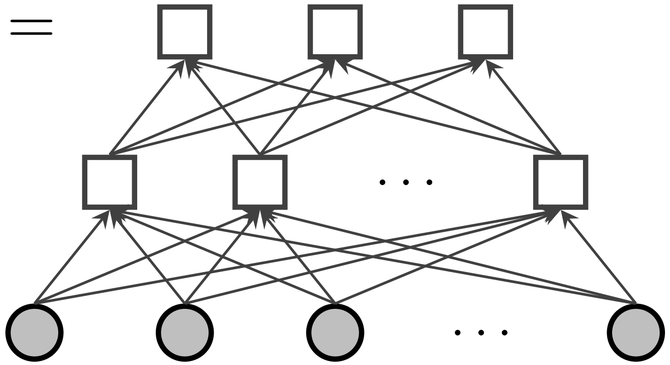
- Parameterize ***f*** with a (deep) neural network.

pros:

- shared representations
- channel external knowledge (e.g. word embeddings)
- high accuracy

cons:

$$\mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m) =$$



Neural sequence labeling

- Parameterize ***f*** with a (deep) neural network.

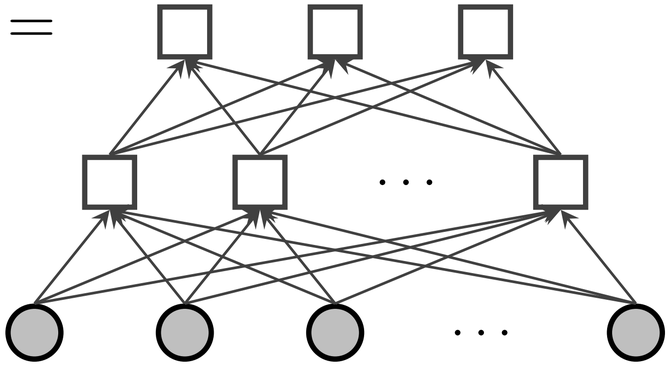
pros:

- shared representations
- channel external knowledge (e.g. word embeddings)
- high accuracy

cons:

- hard to interpret feature meaning, explain predictions
- optimization/hyperparameters
- prone to overfitting
- compute-heavy

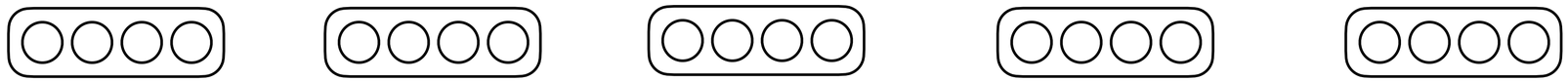
$$\mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m) =$$



Neural sequence labeling

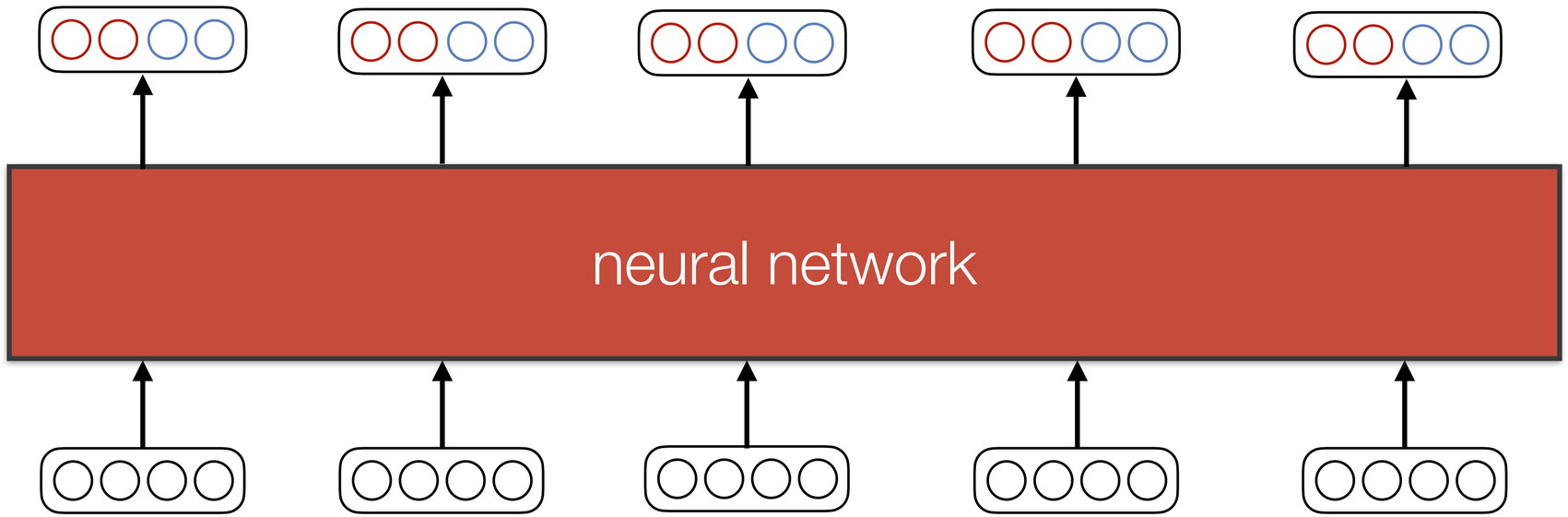
Neural sequence labeling

word embeddings

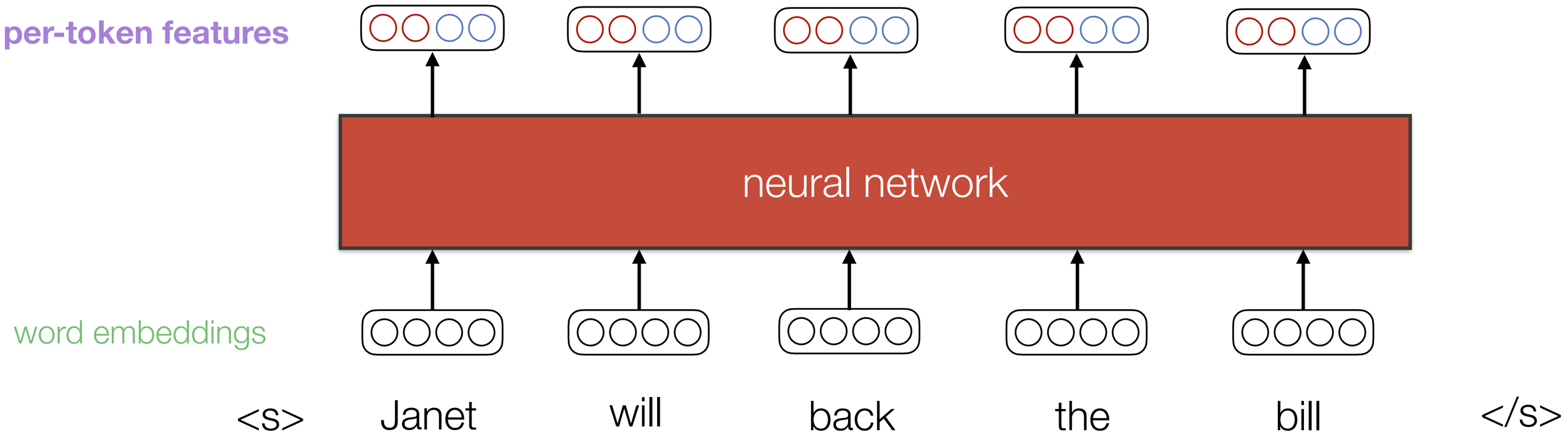


<s> Janet will back the bill </s>

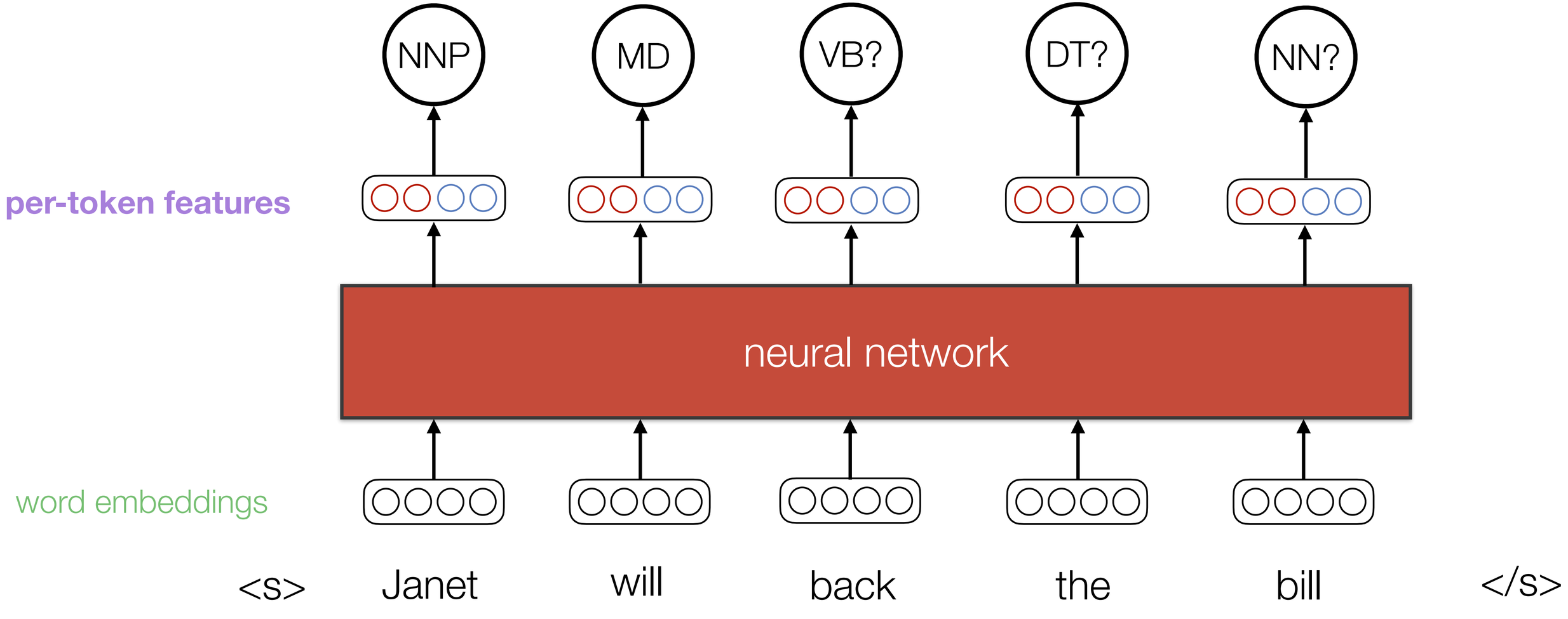
Neural sequence labeling



Neural sequence labeling

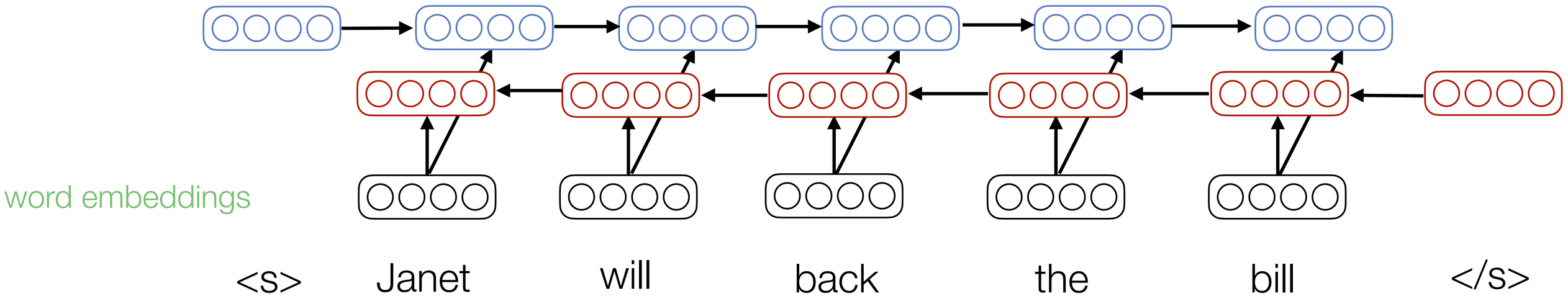


Neural sequence labeling



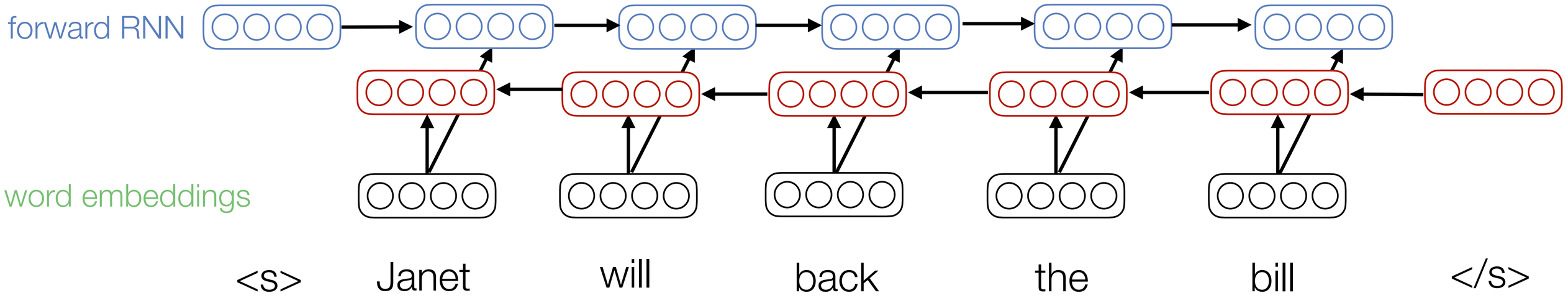
Neural sequence labeling

Bidirectional RNNs



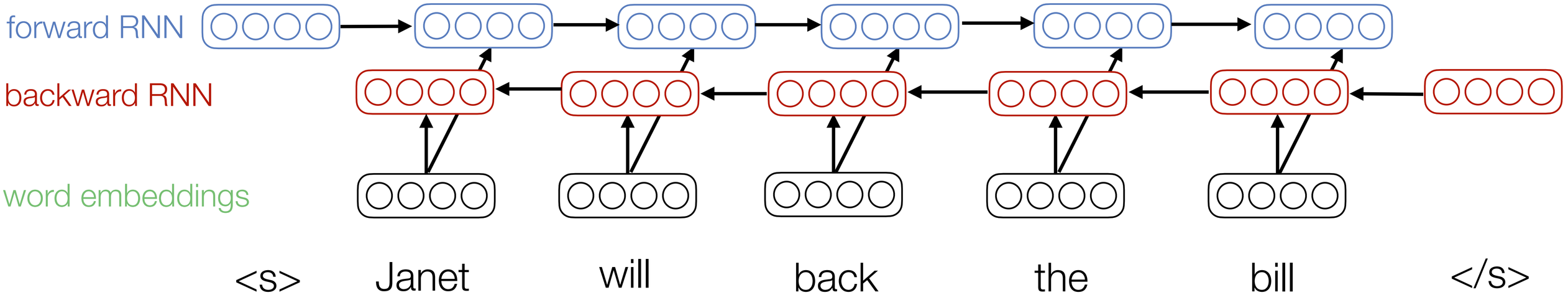
Neural sequence labeling

Bidirectional RNNs



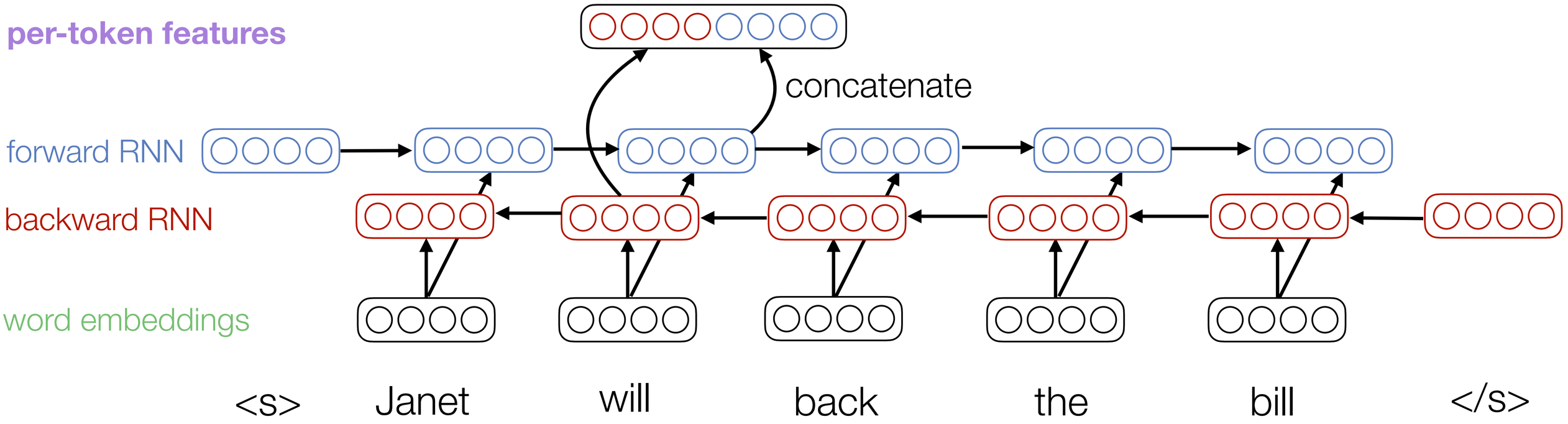
Neural sequence labeling

Bidirectional RNNs



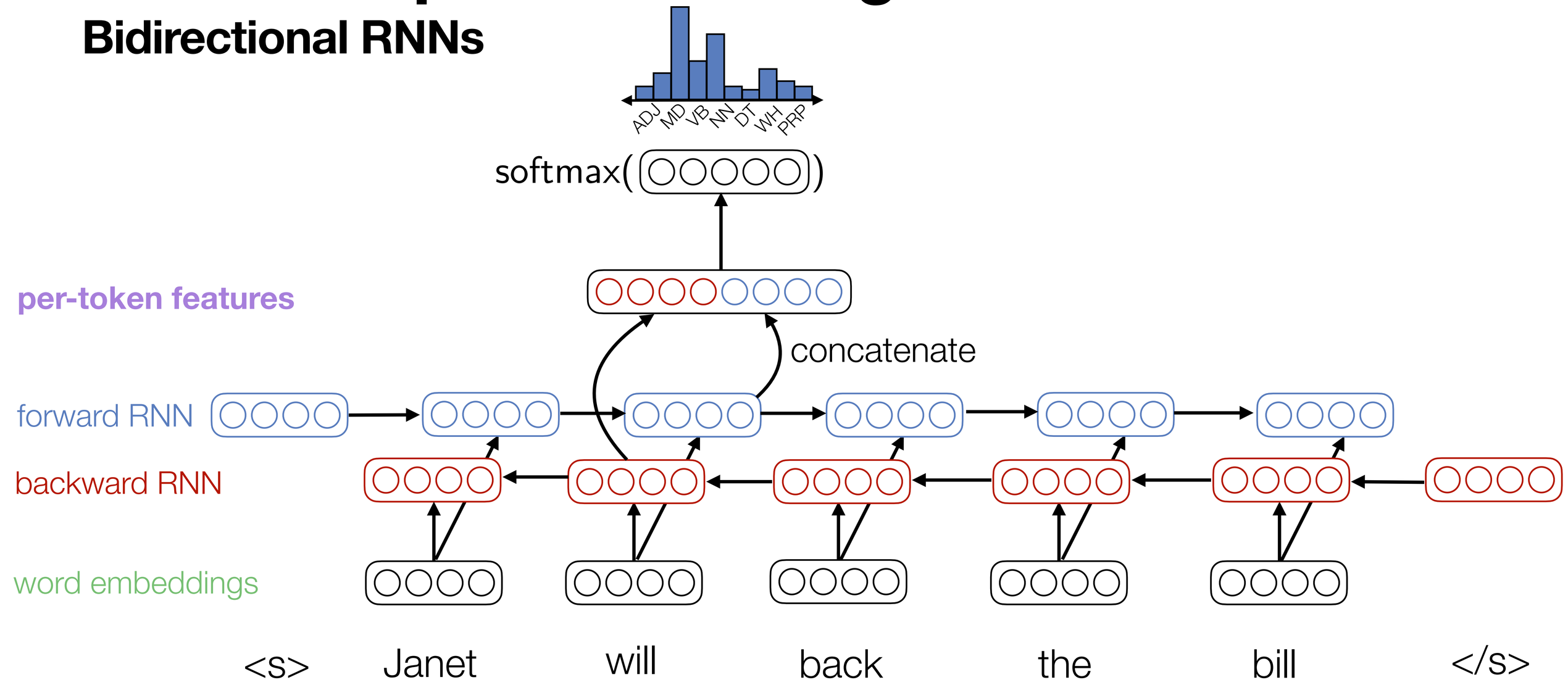
Neural sequence labeling

Bidirectional RNNs



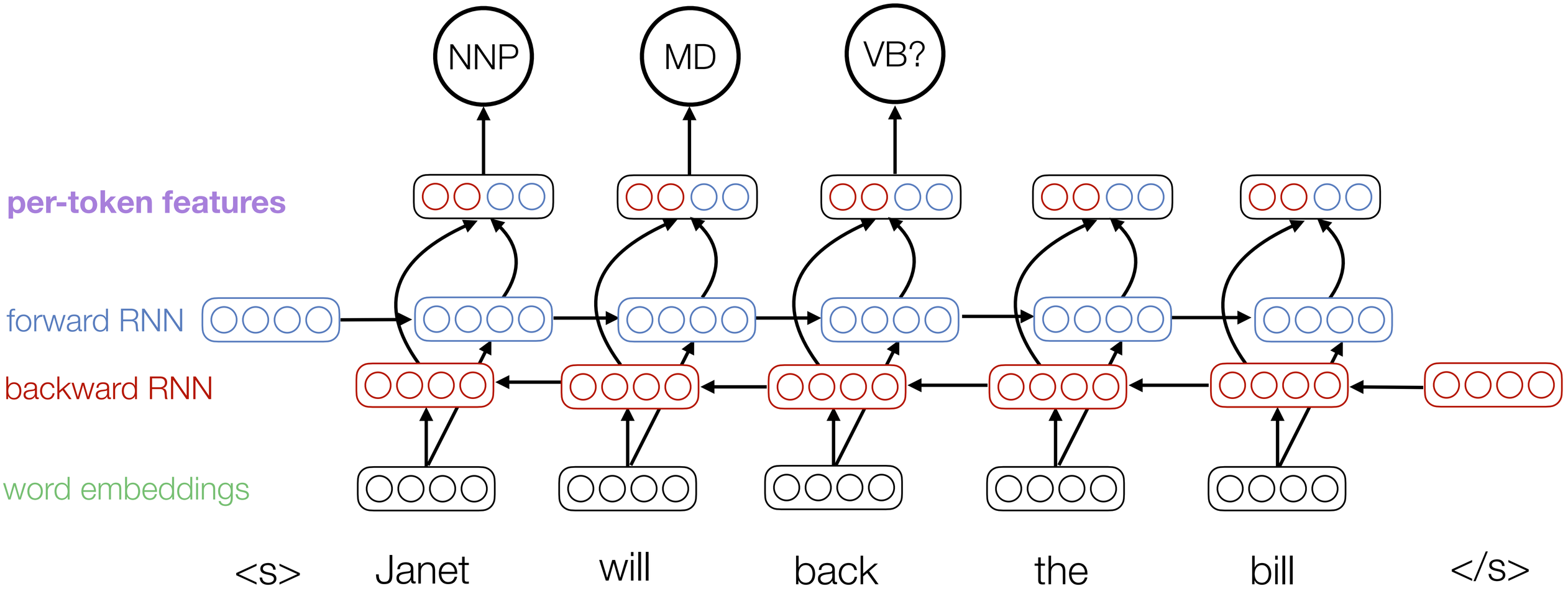
Neural sequence labeling

Bidirectional RNNs



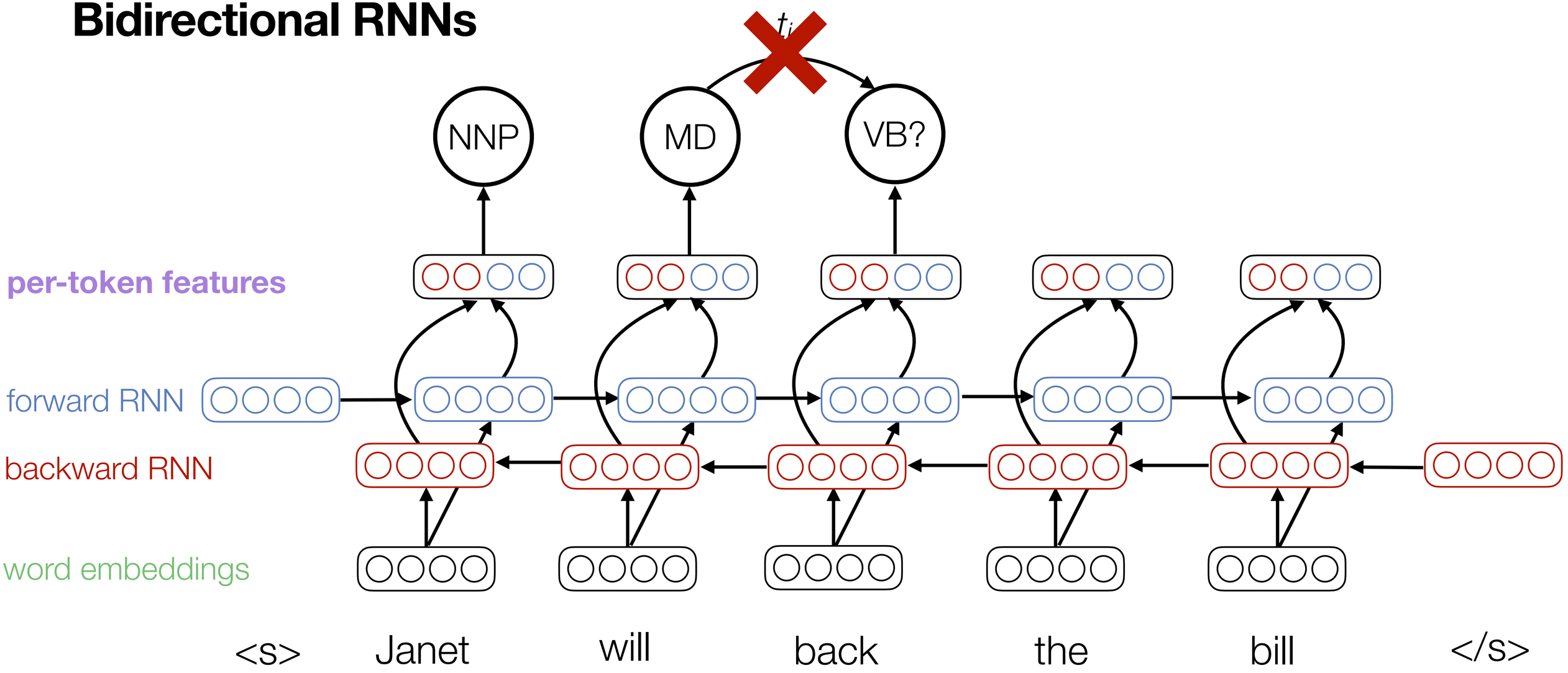
Neural sequence labeling

Bidirectional RNNs



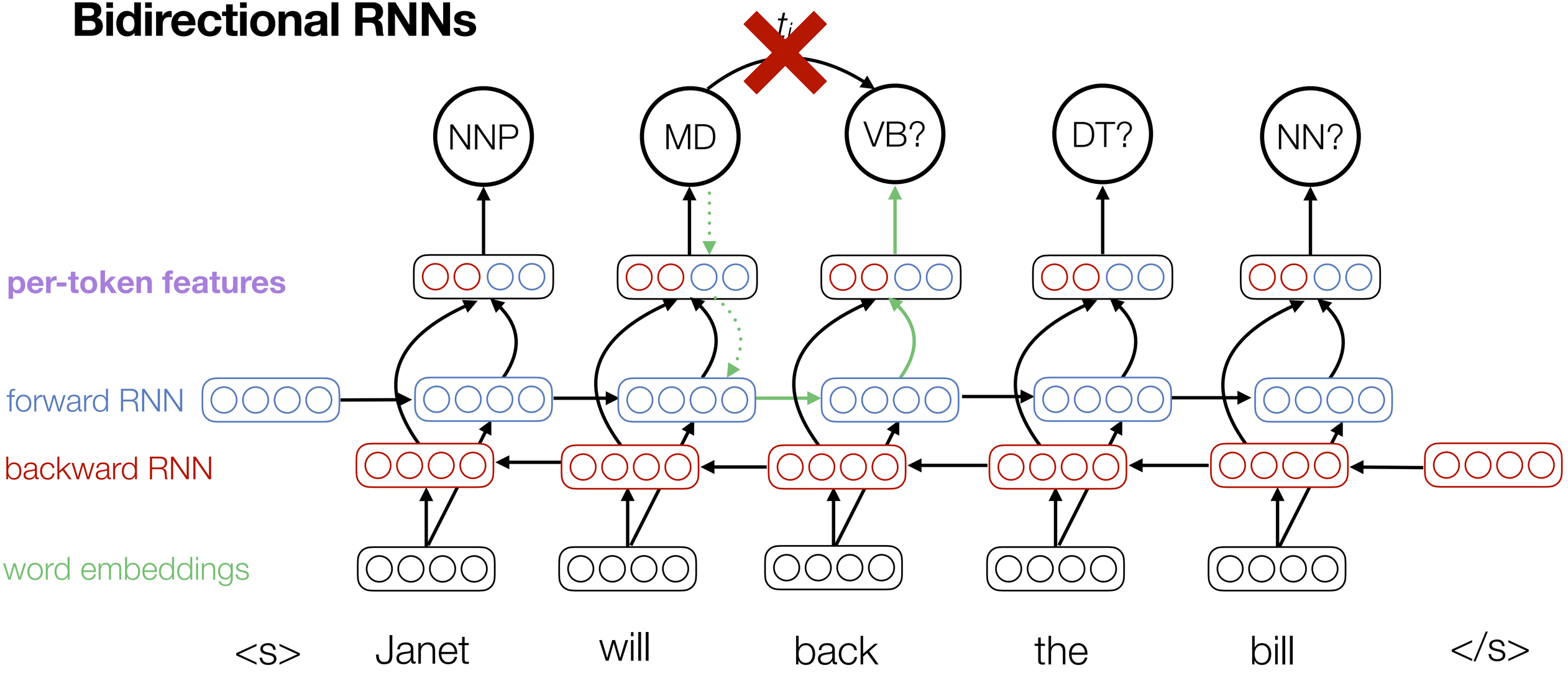
Neural sequence labeling

Bidirectional RNNs



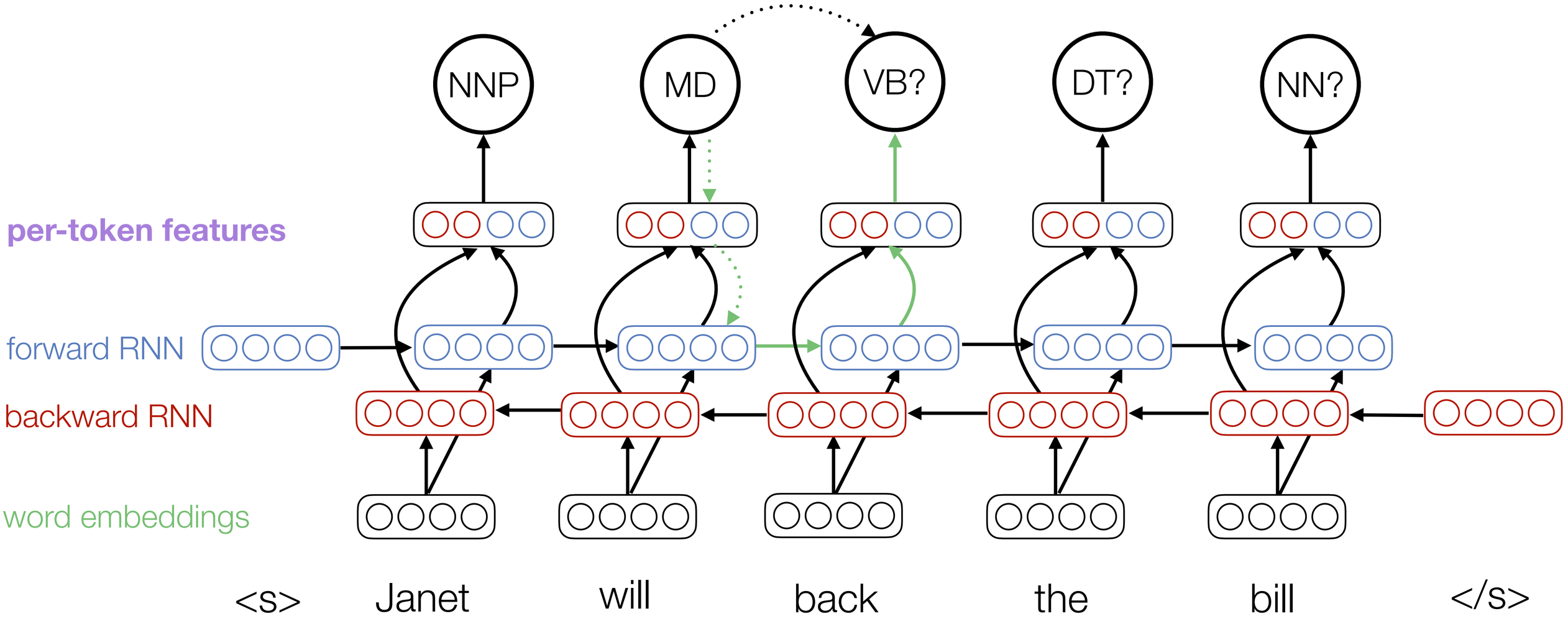
Neural sequence labeling

Bidirectional RNNs



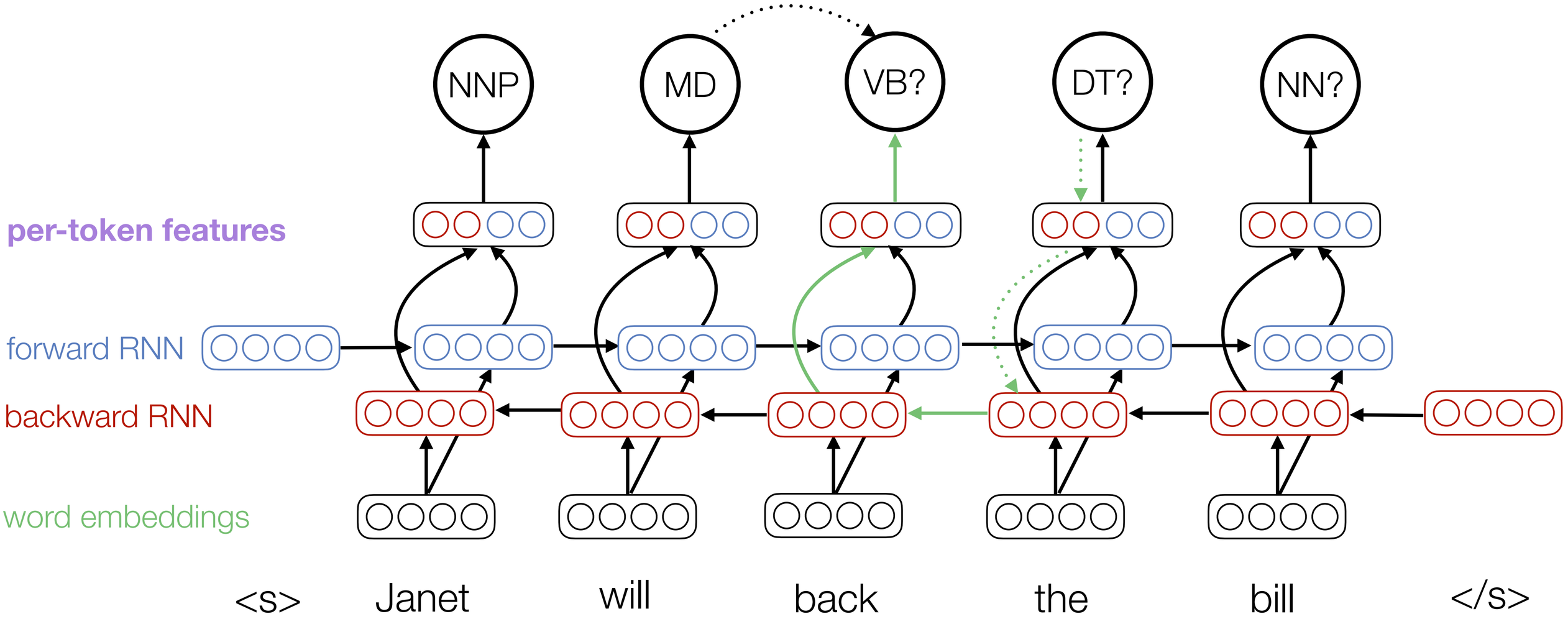
Neural sequence labeling

Bidirectional RNNs



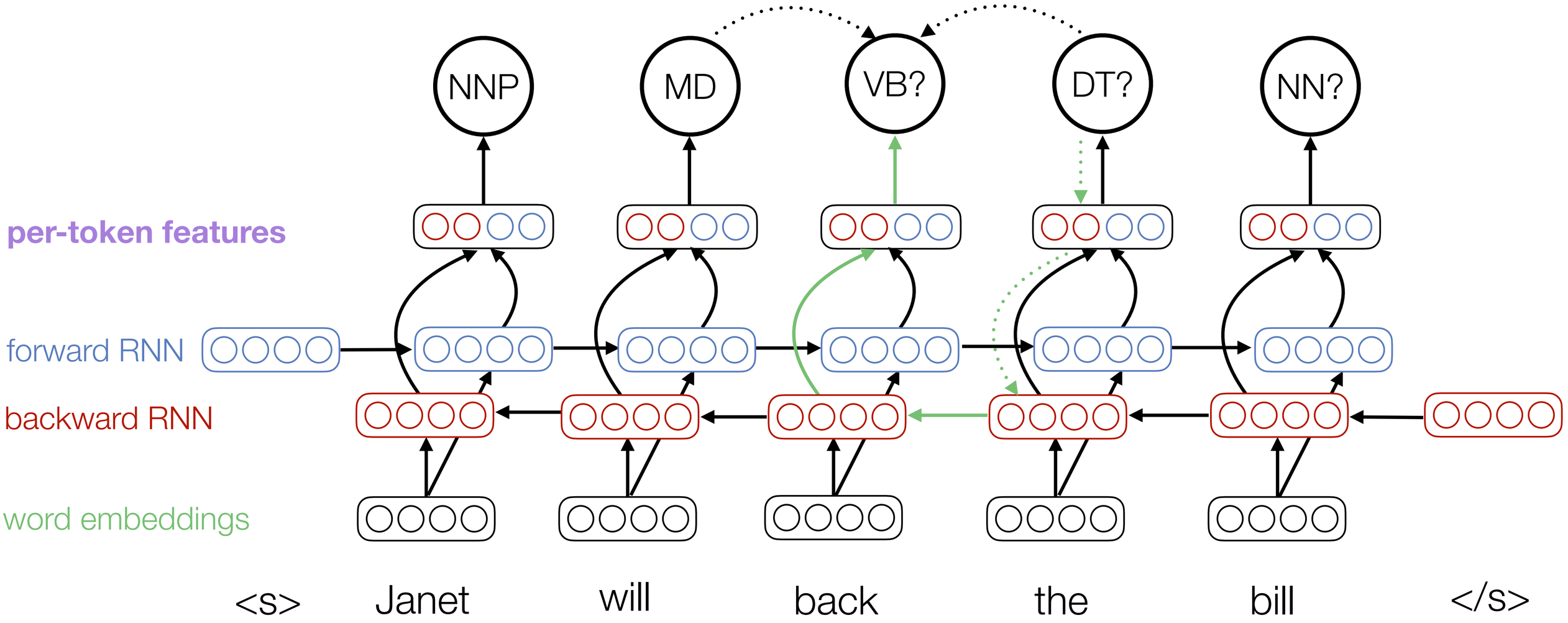
Neural sequence labeling

Bidirectional RNNs



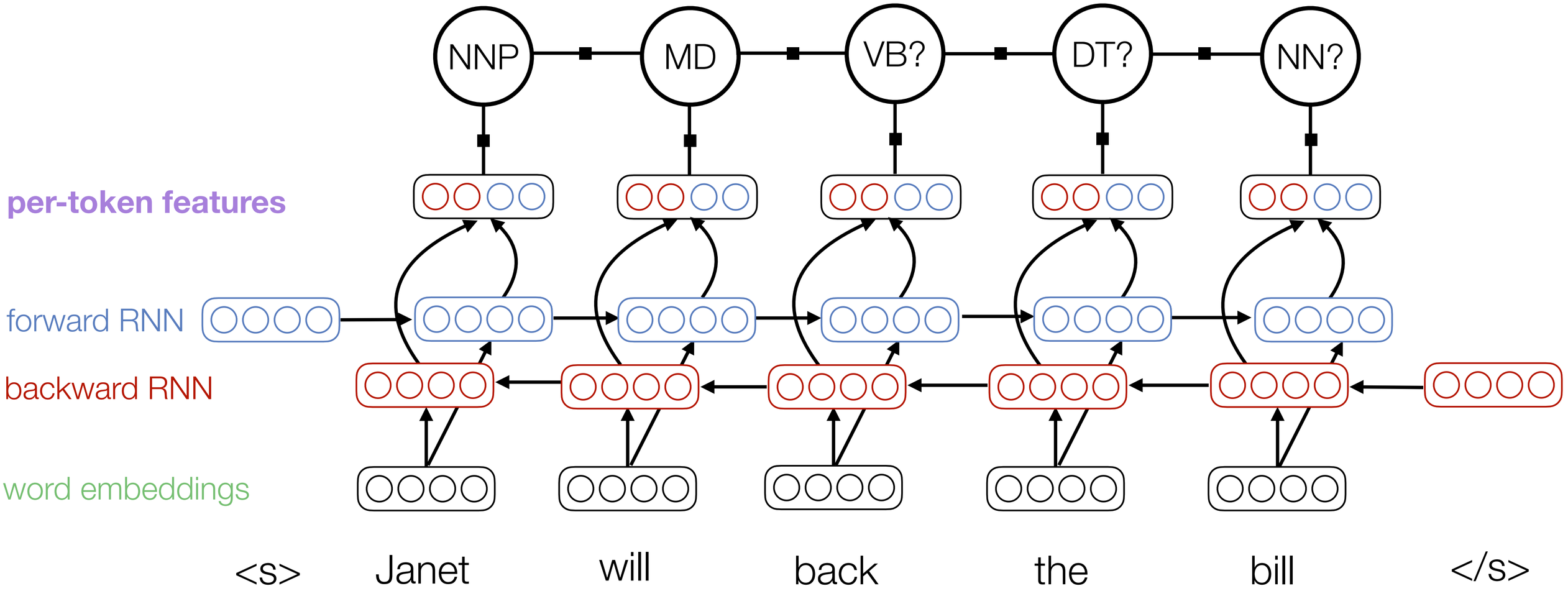
Neural sequence labeling

Bidirectional RNNs

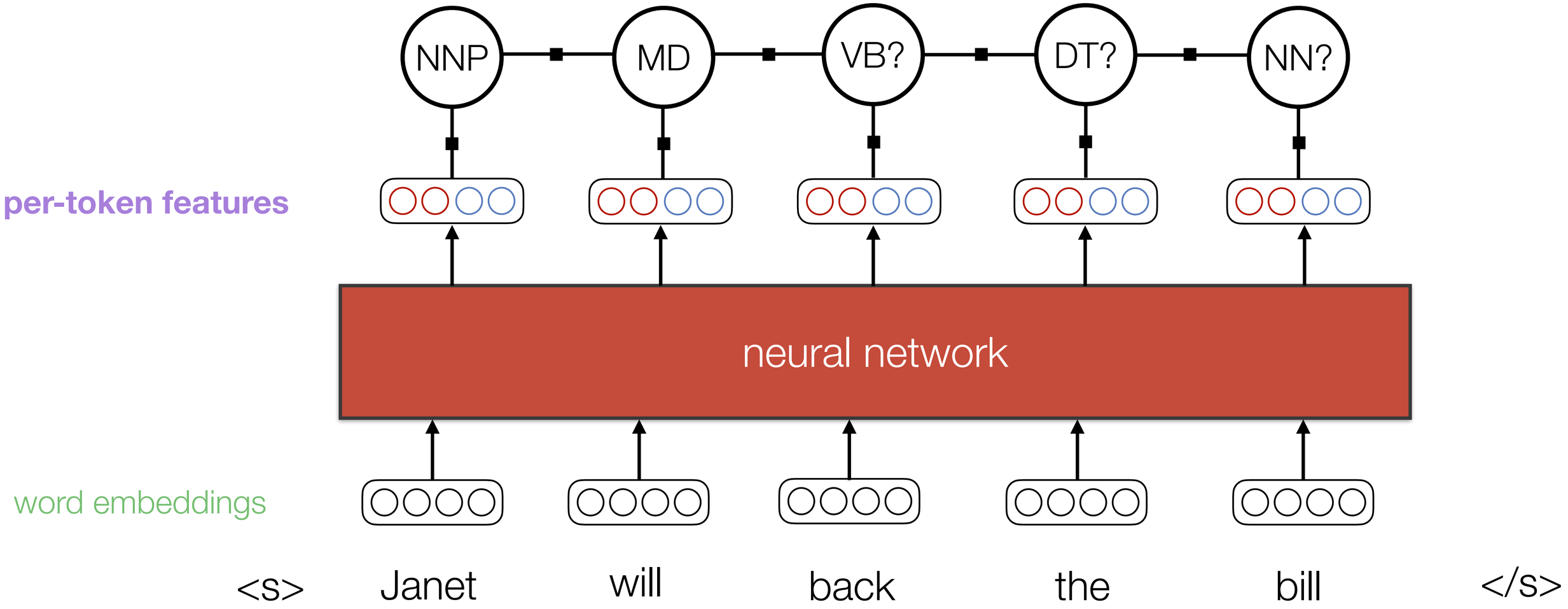


Neural sequence labeling

Bidirectional RNN-CRFs



Neural sequence labeling



Convolutional neural networks

Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.

Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.
- Unlike computer vision, in NLP we use 1D CNNs.

Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.
- Unlike computer vision, in NLP we use 1D CNNs.
- For sentence/document classification: **pooling function** over representations.

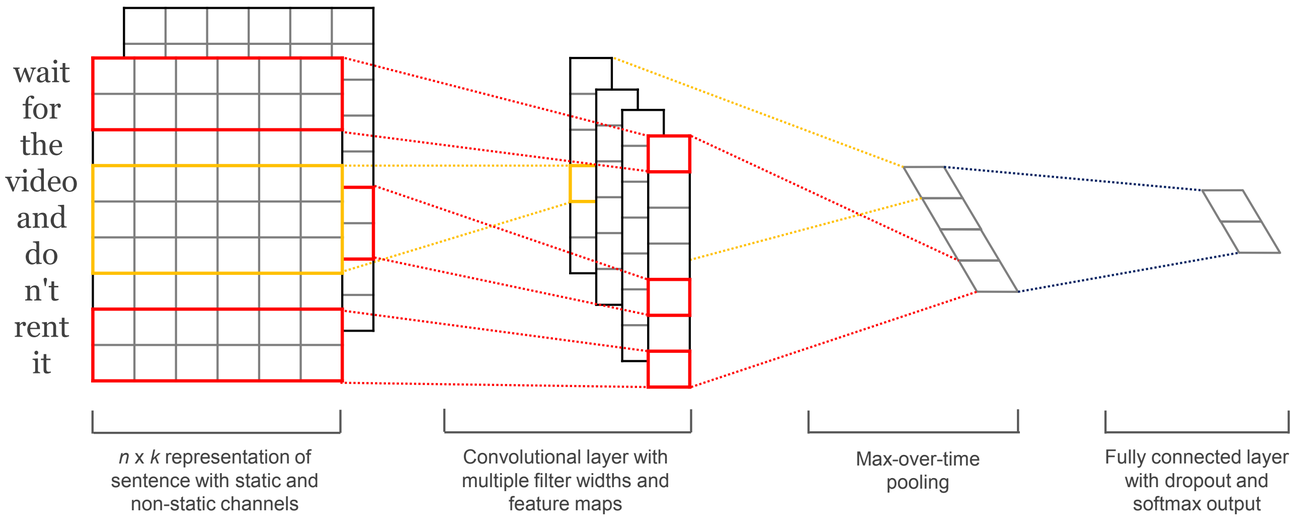


Figure from: Yoon Kim. Convolutional Neural Networks for Sentence Classification. EMNLP 2014.

Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.
- Unlike computer vision, in NLP we use 1D CNNs.
- For sentence/document classification: **pooling function** over representations.
  - For example: sum, average. Most common: **max pooling** (over time).

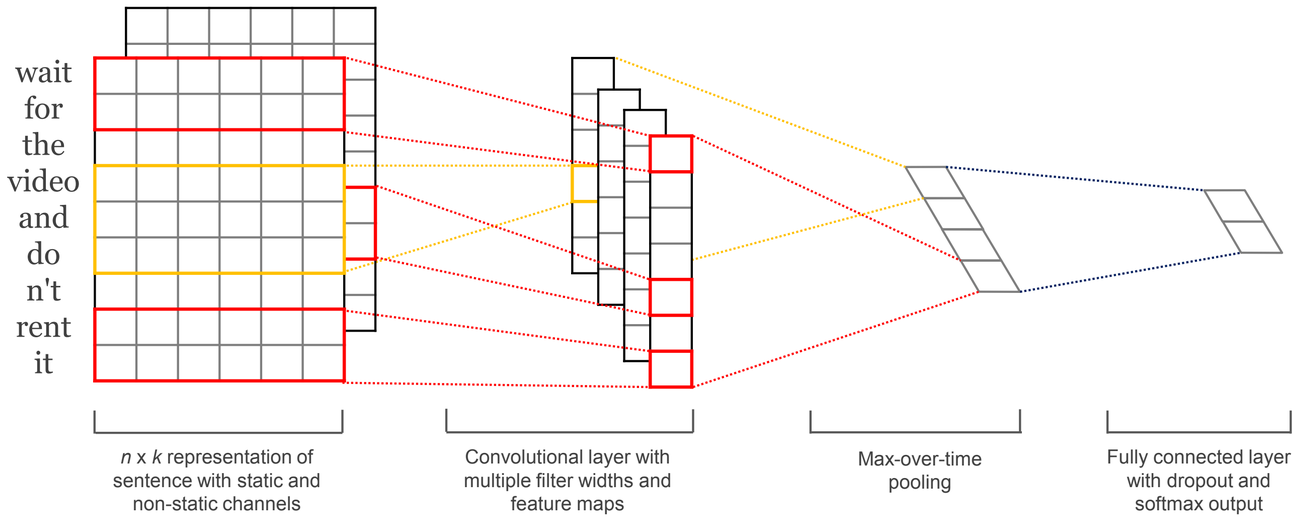
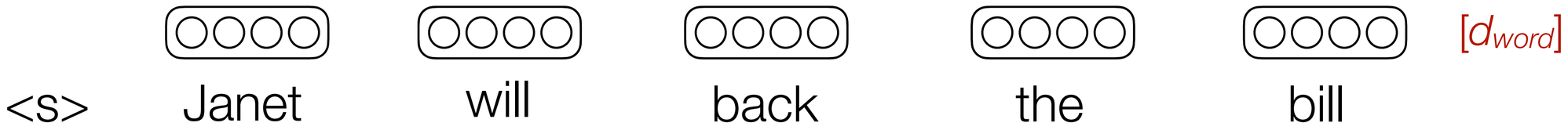


Figure from: Yoon Kim. Convolutional Neural Networks for Sentence Classification. EMNLP 2014.

Convolutional neural networks

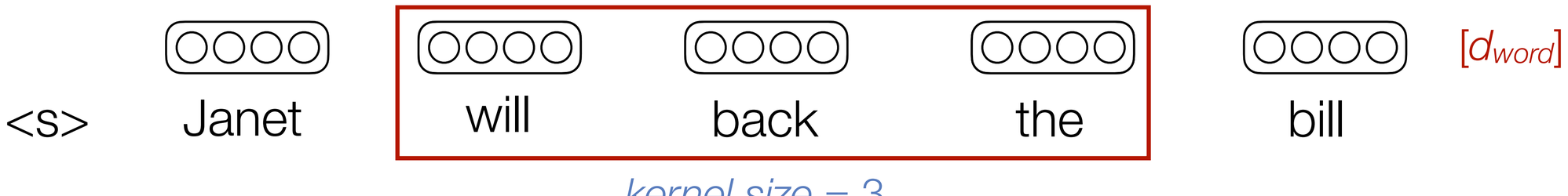
- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.

dims:



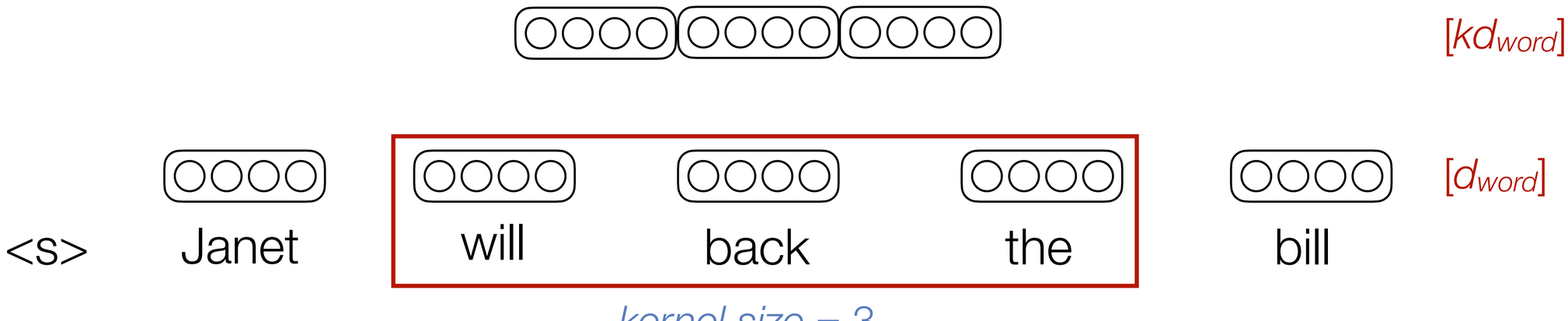
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



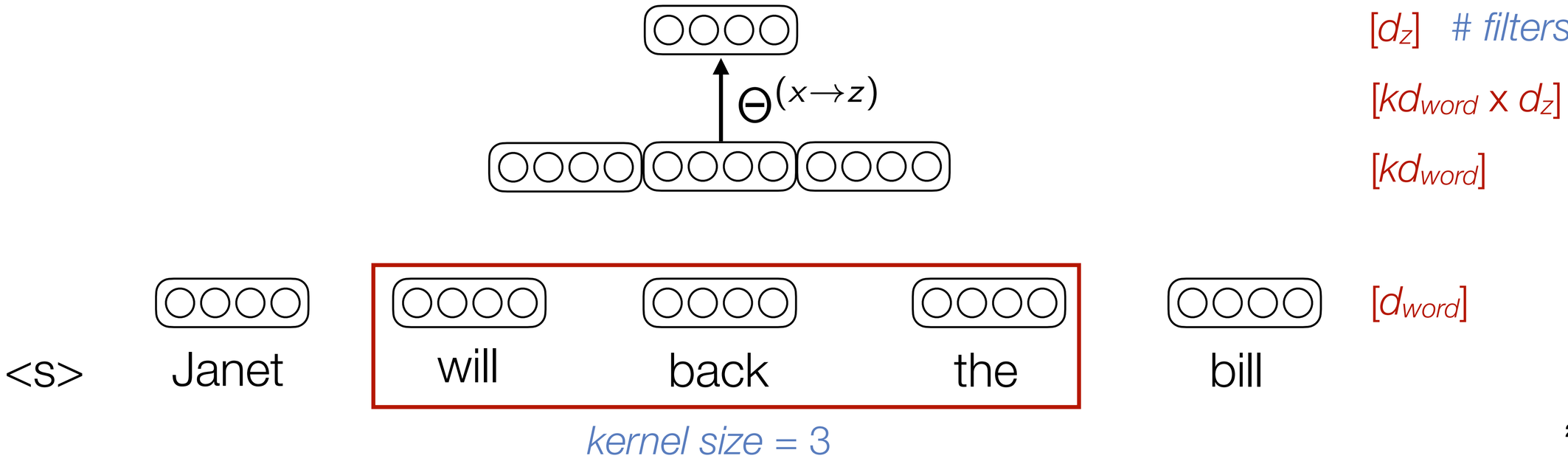
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



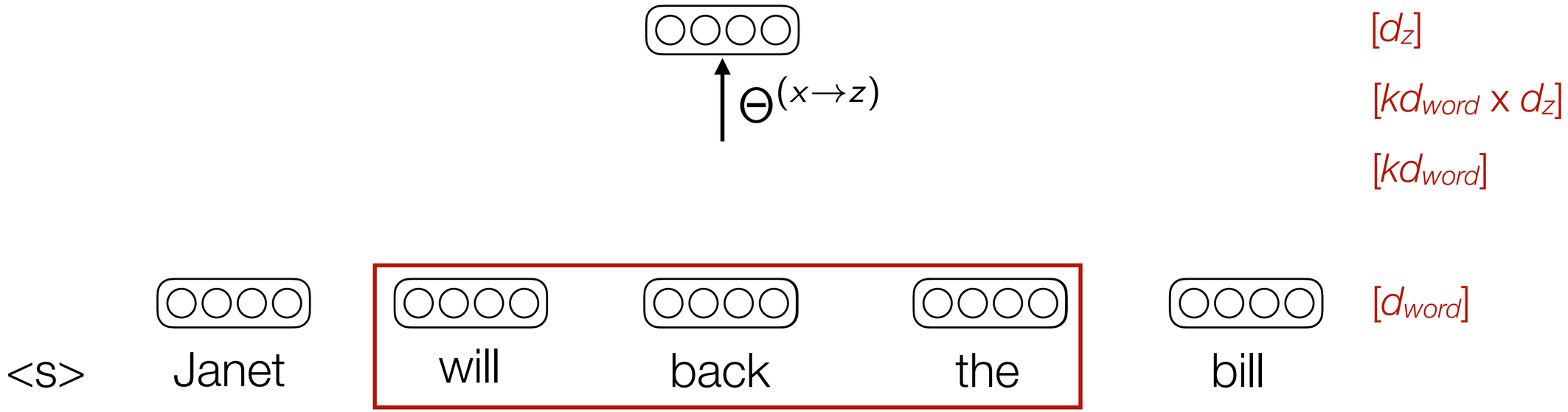
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



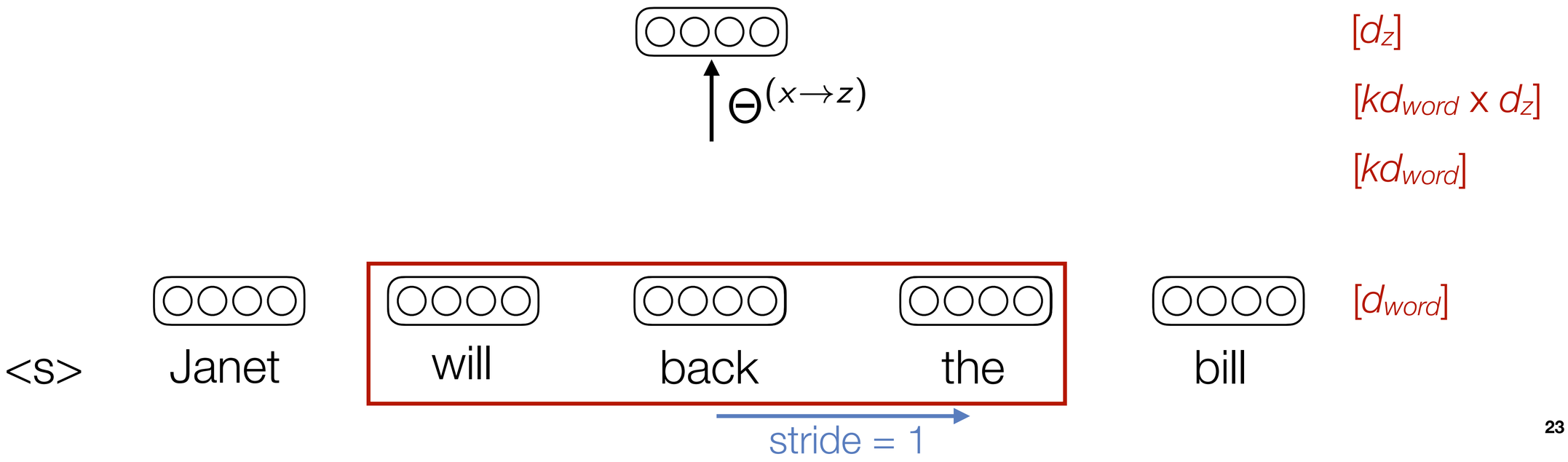
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



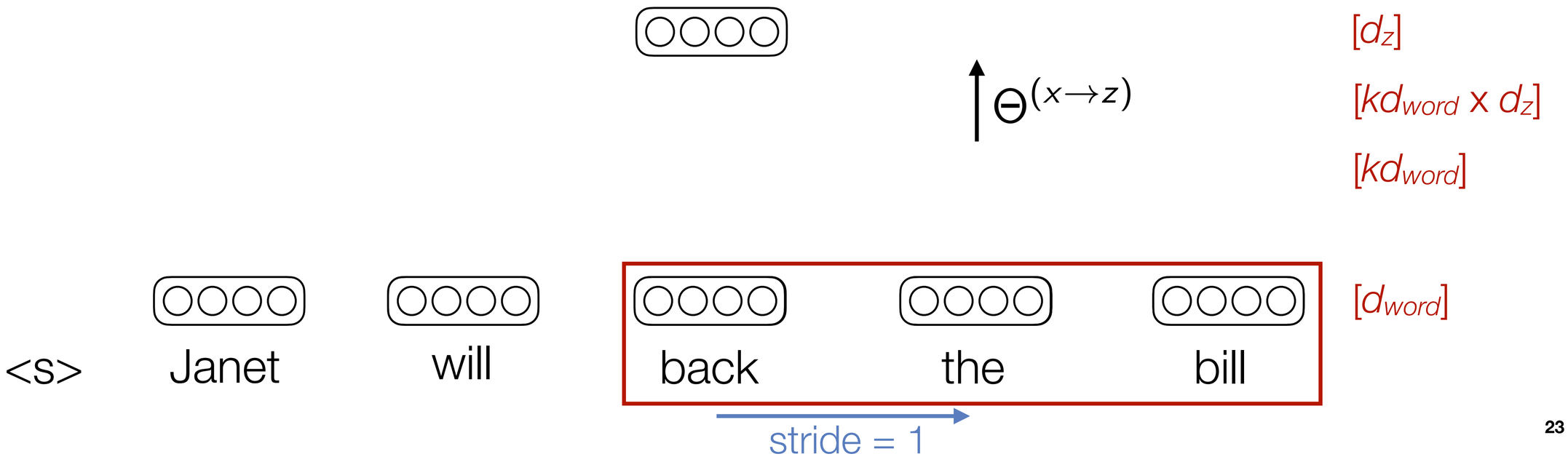
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



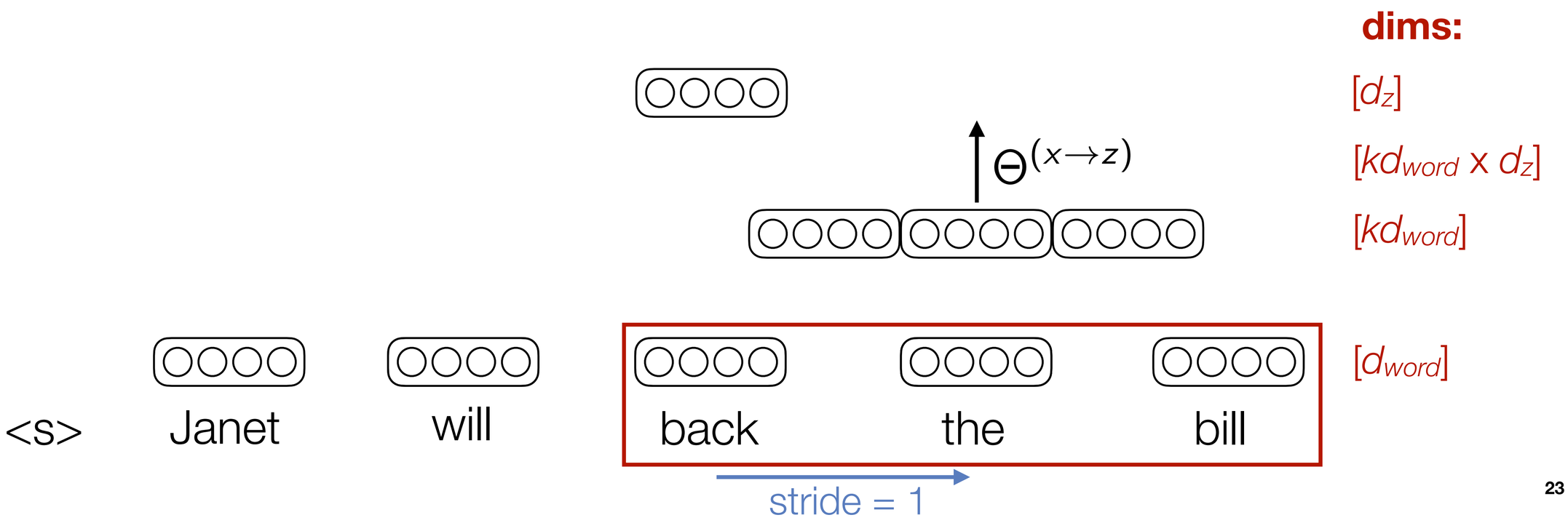
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



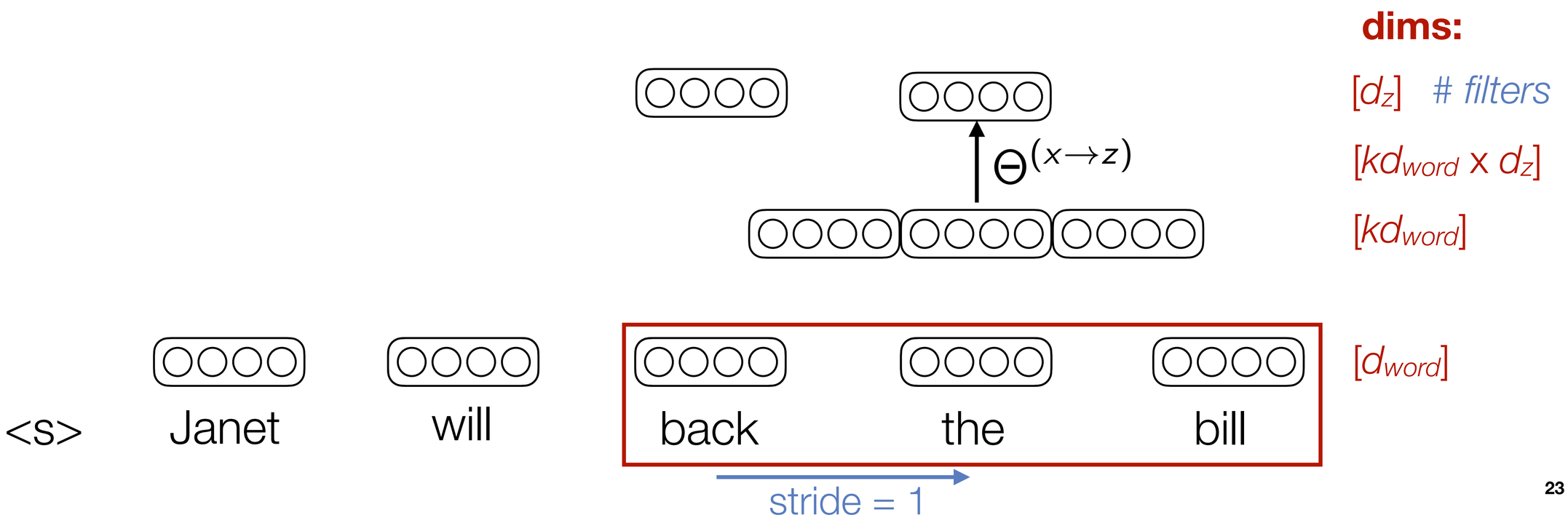
Convolutional neural networks

- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.

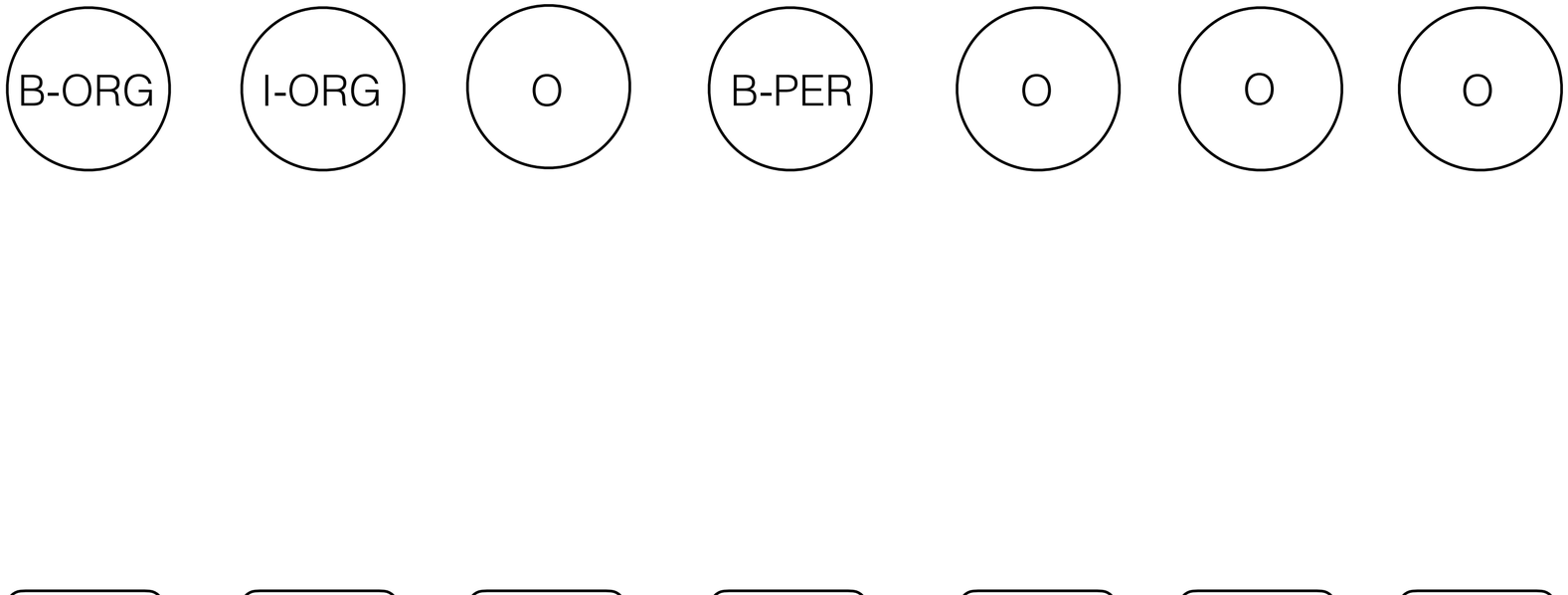


Convolutional neural networks

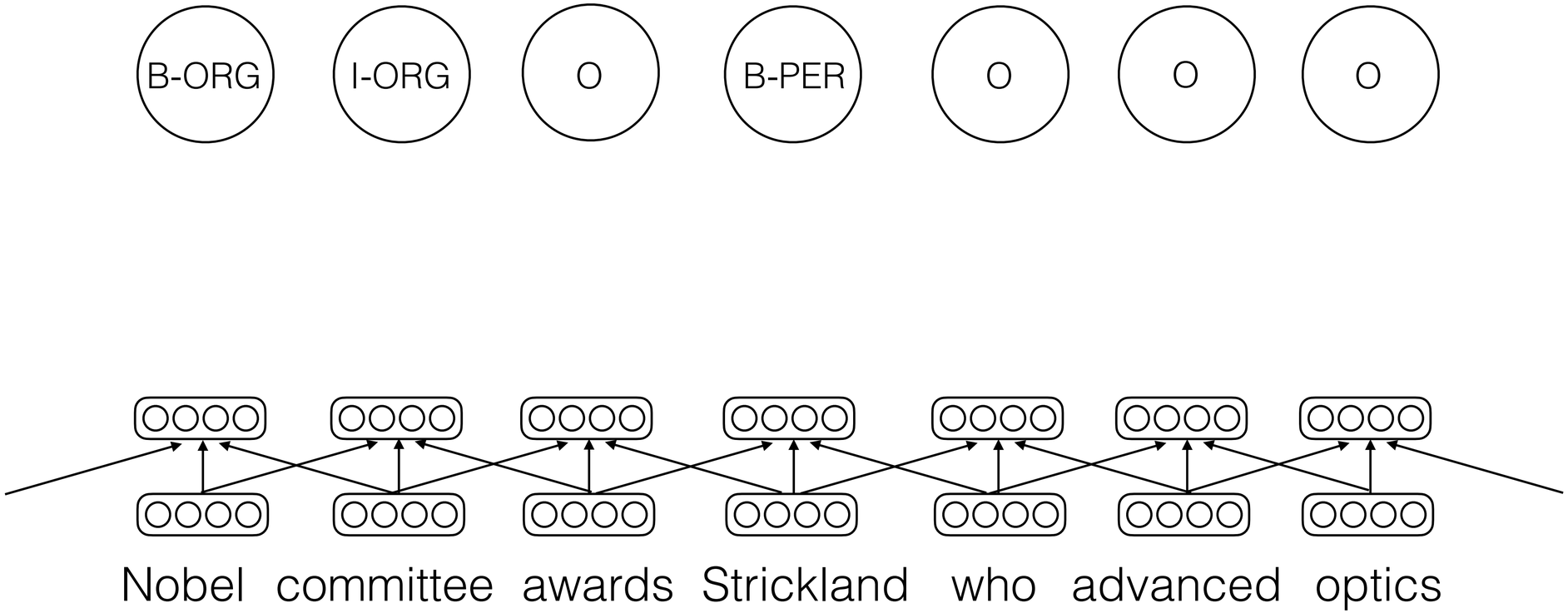
- In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.



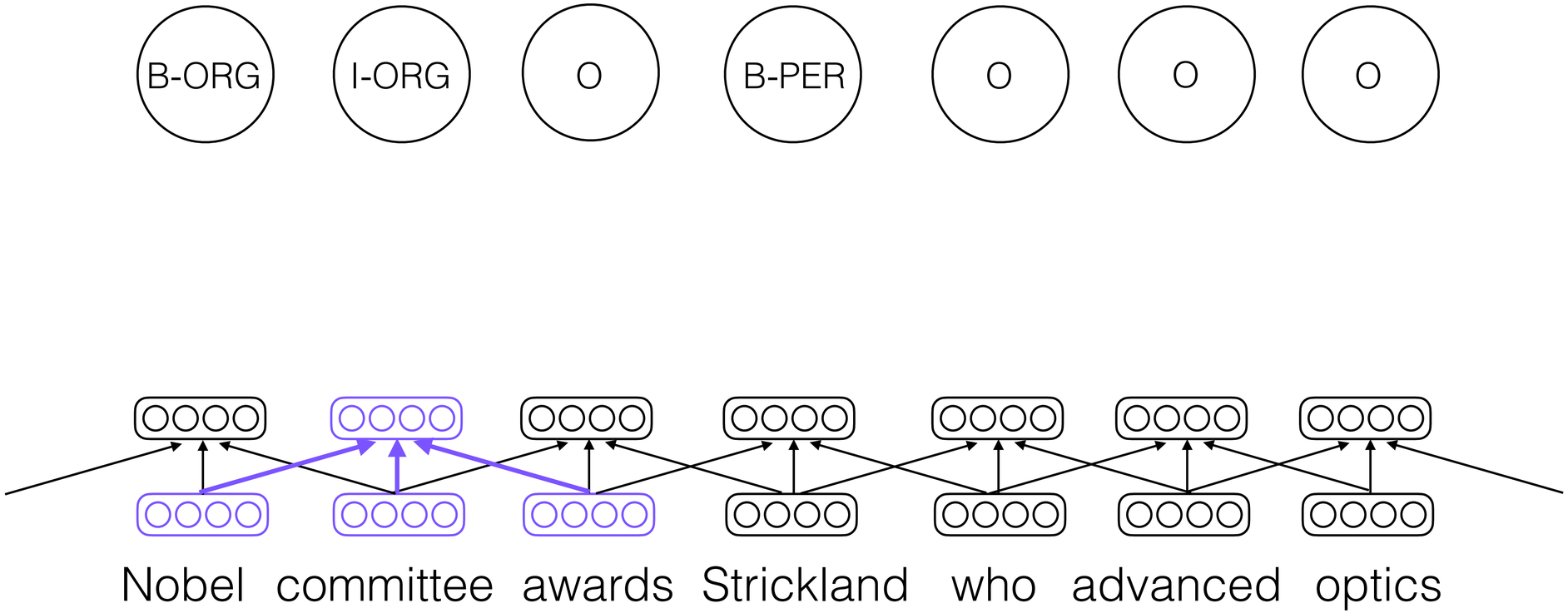
Sequence labeling w/ CNNs



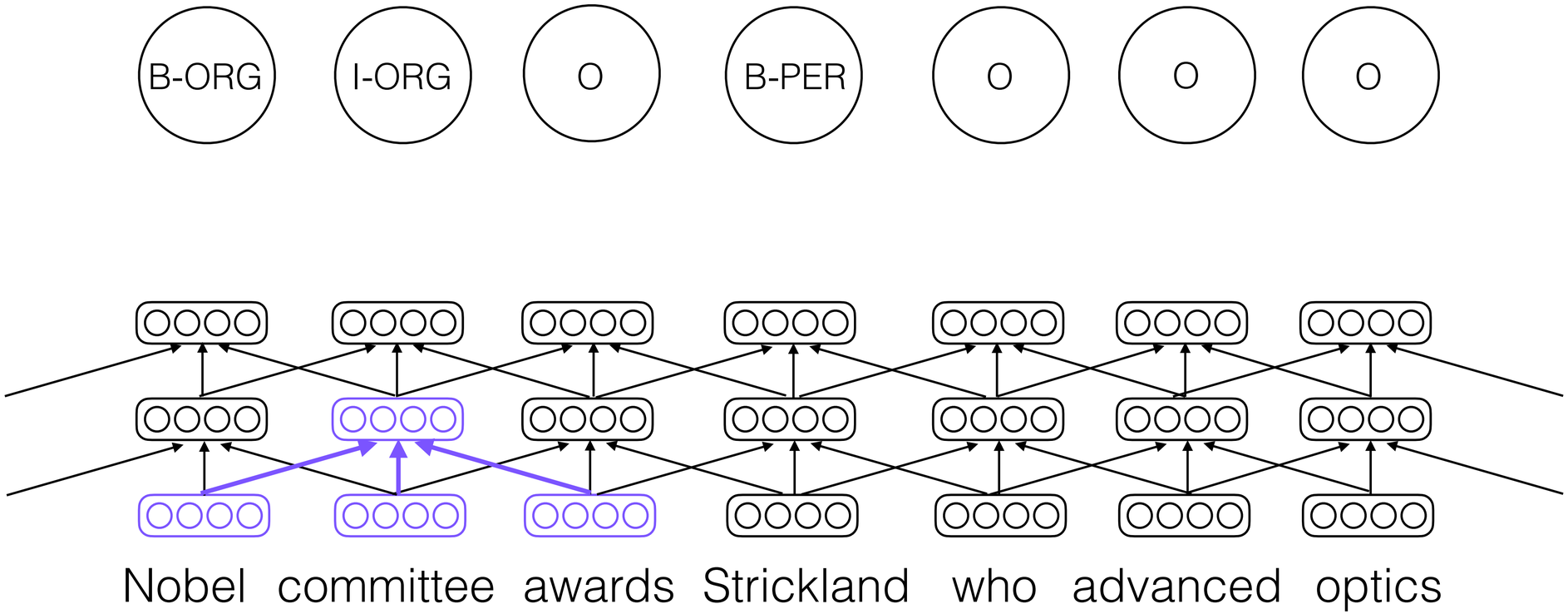
Sequence labeling w/ CNNs



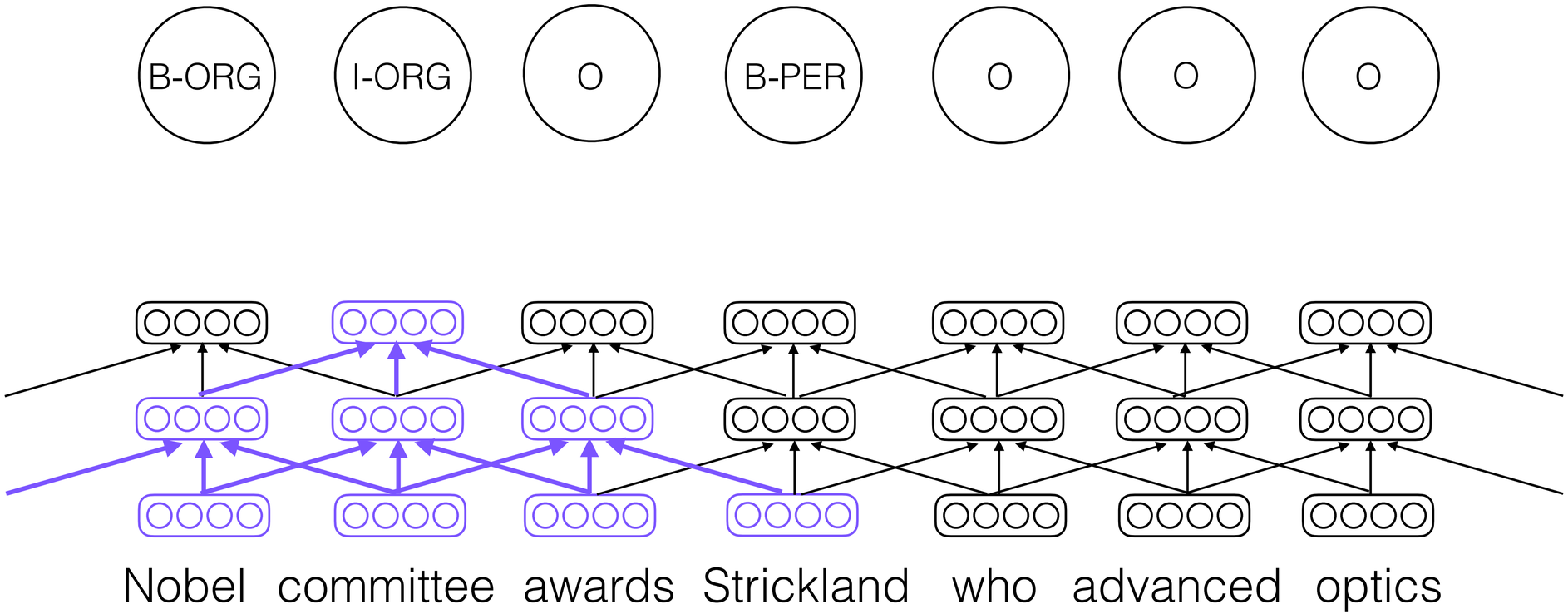
Sequence labeling w/ CNNs



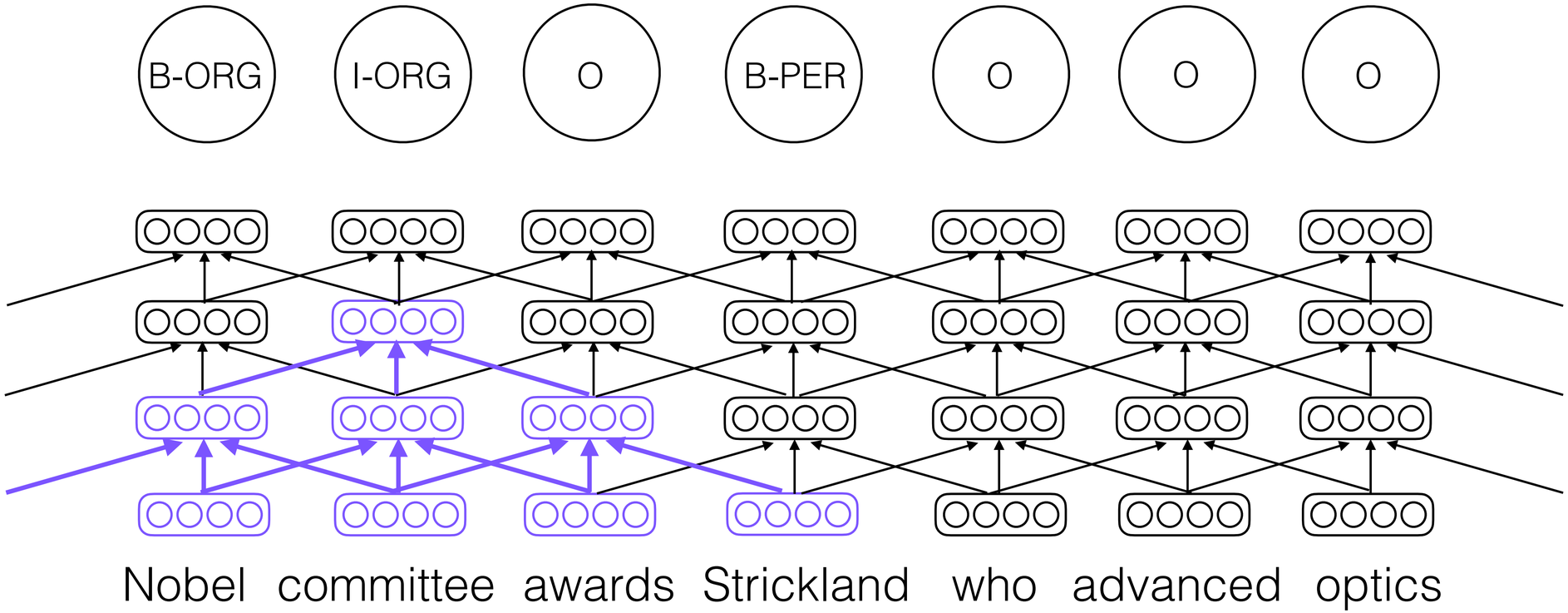
Sequence labeling w/ CNNs



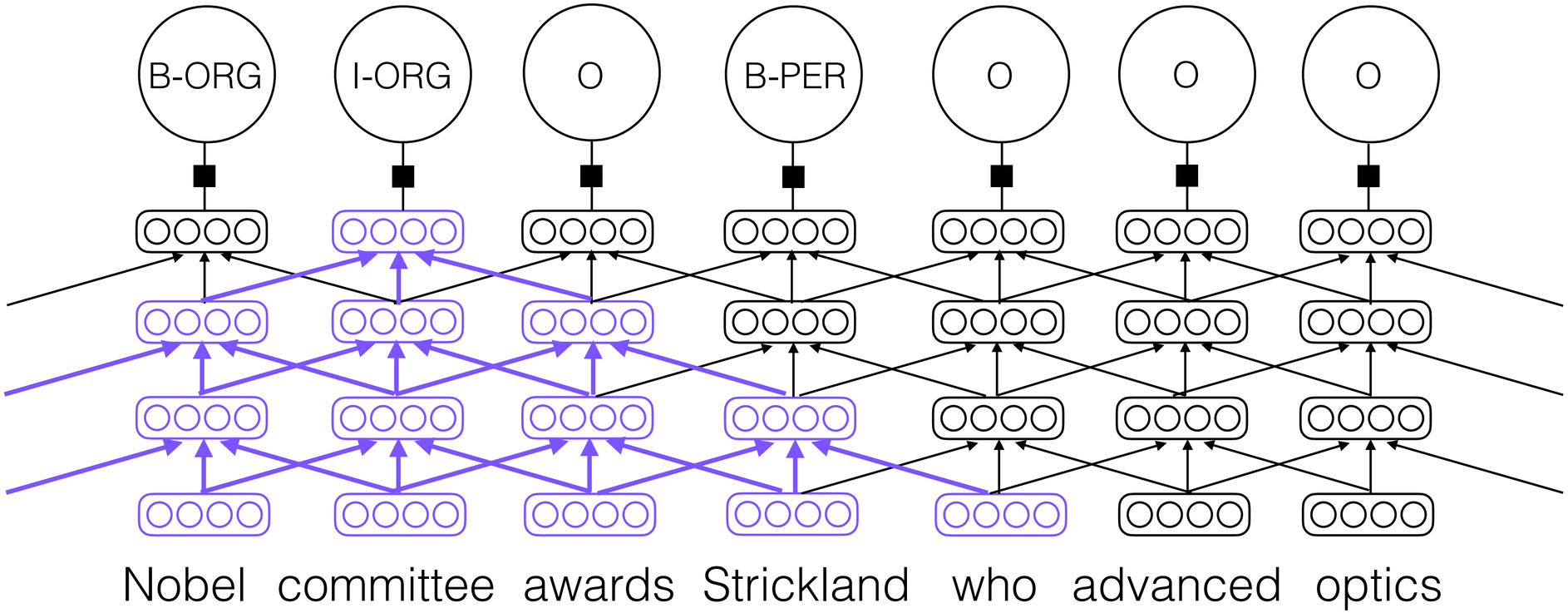
Sequence labeling w/ CNNs



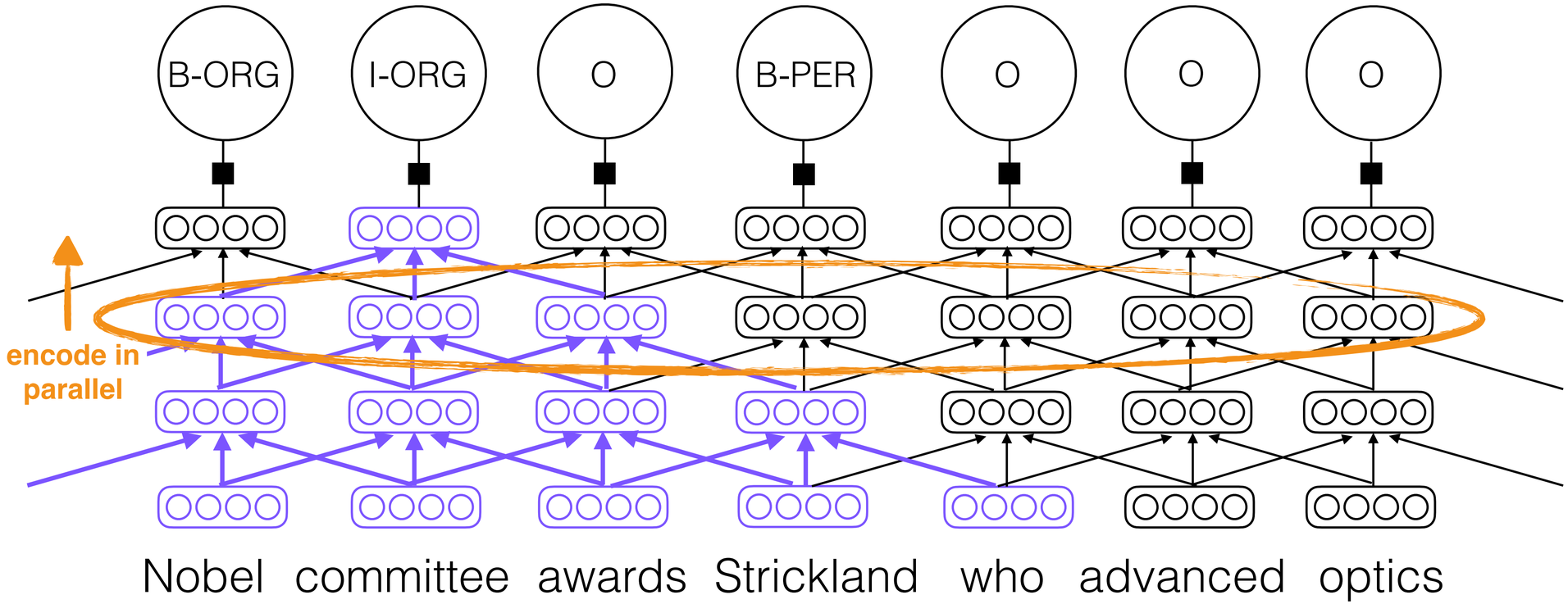
Sequence labeling w/ CNNs



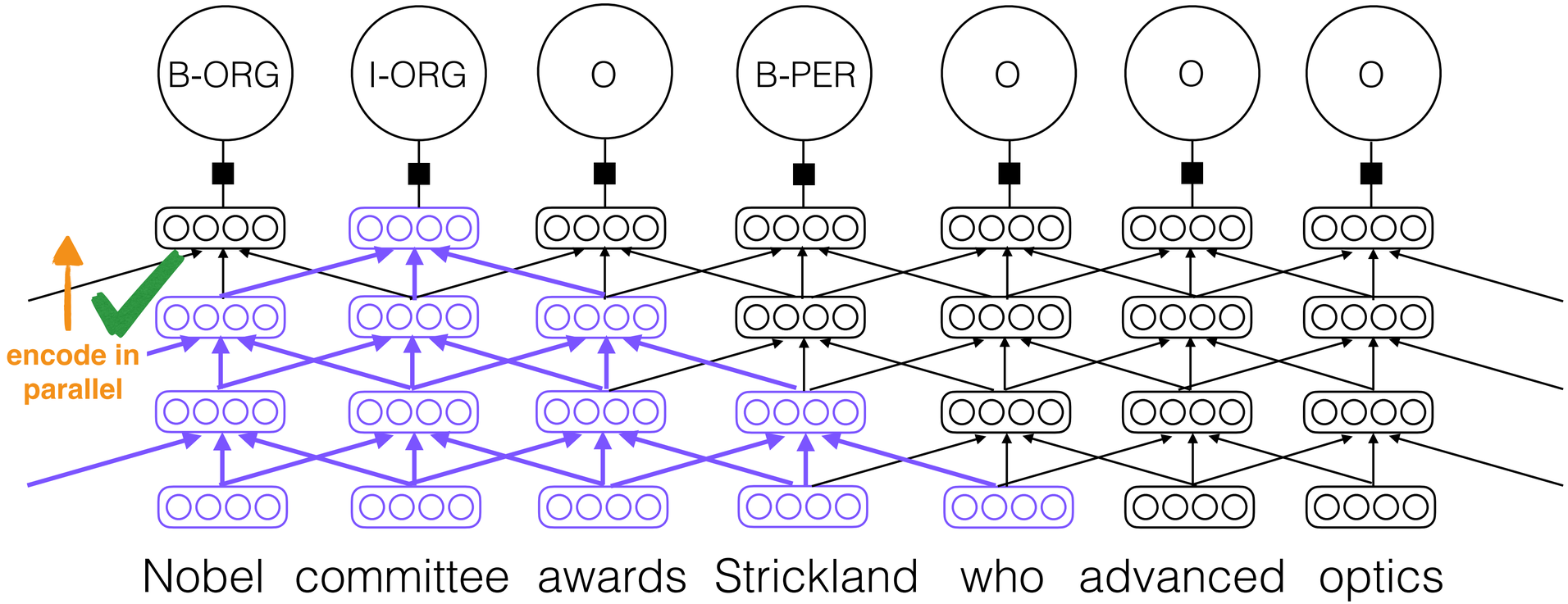
Sequence labeling w/ CNNs



Sequence labeling w/ CNNs

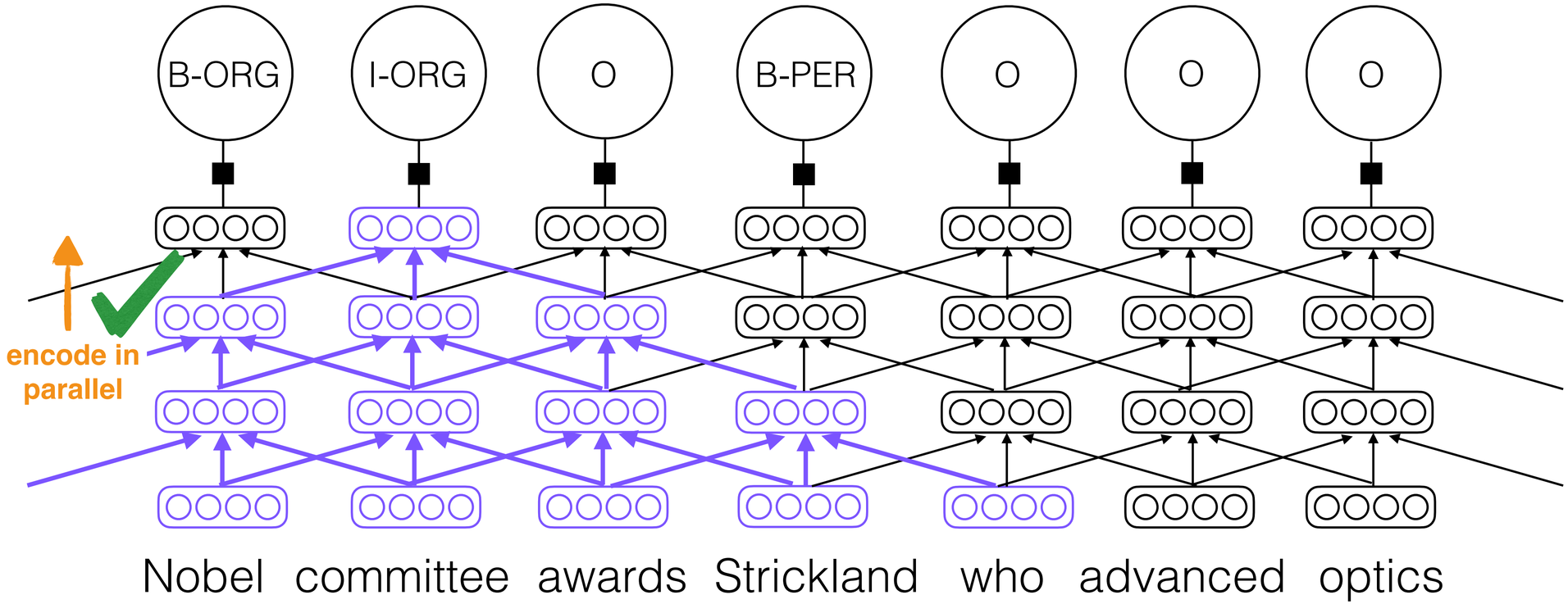


Sequence labeling w/ CNNs



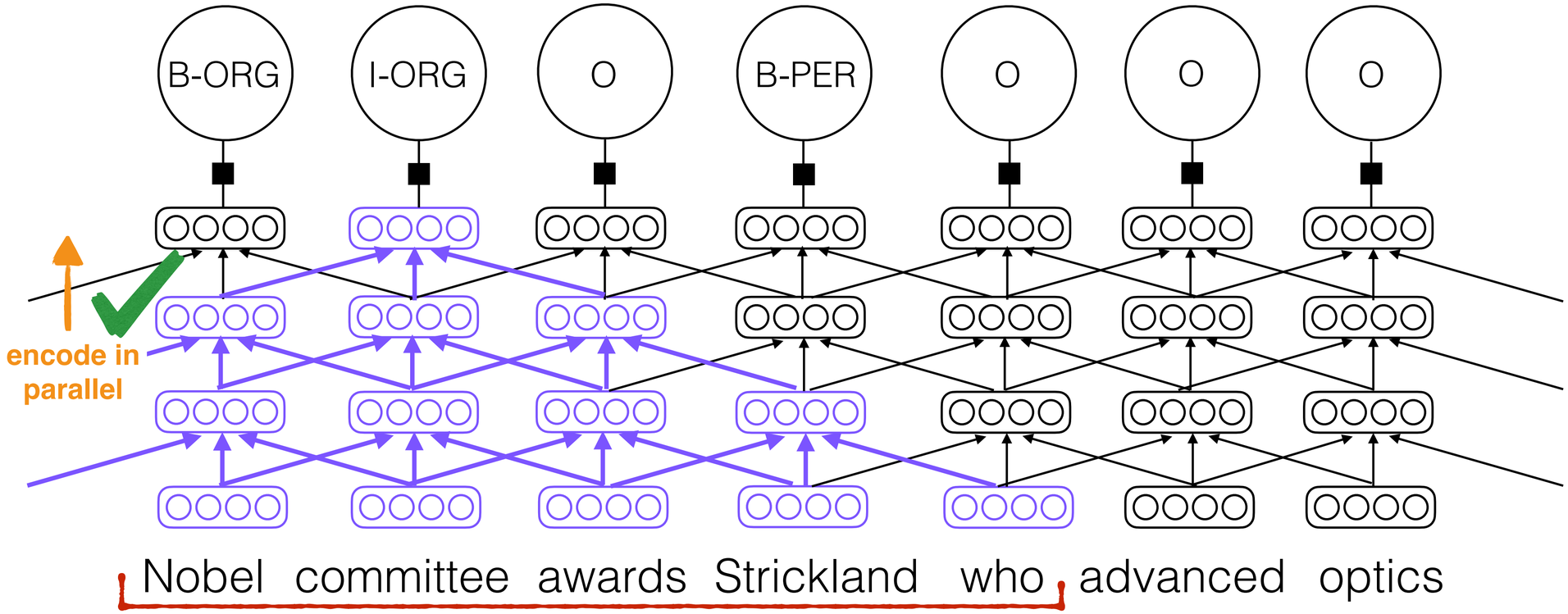
Sequence labeling w/ CNNs

- Used for semantic role labeling, with poor results [Collobert et al. 2011].



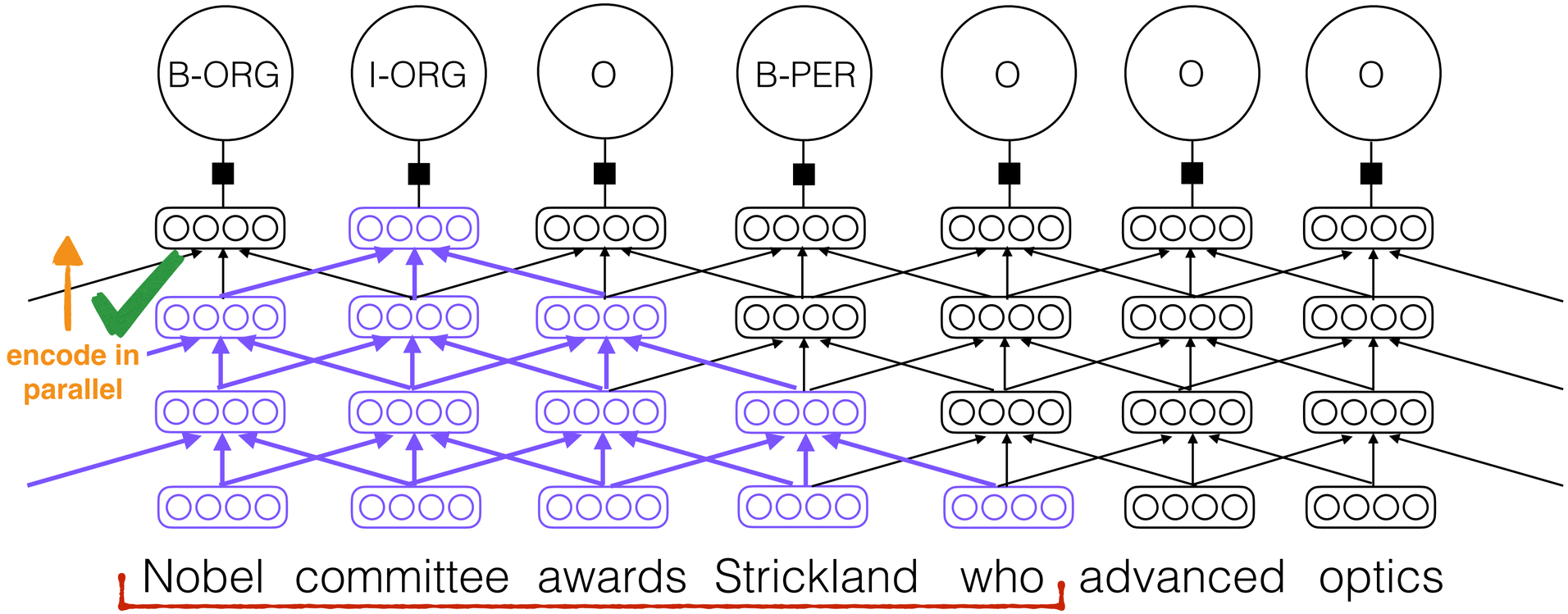
Sequence labeling w/ CNNs

- Used for semantic role labeling, with poor results [Collobert et al. 2011].



Sequence labeling w/ CNNs

- Used for semantic role labeling, with poor results [Collobert et al. 2011].
- Not enough context: amount of context grows **linearly** w/ number of layers.



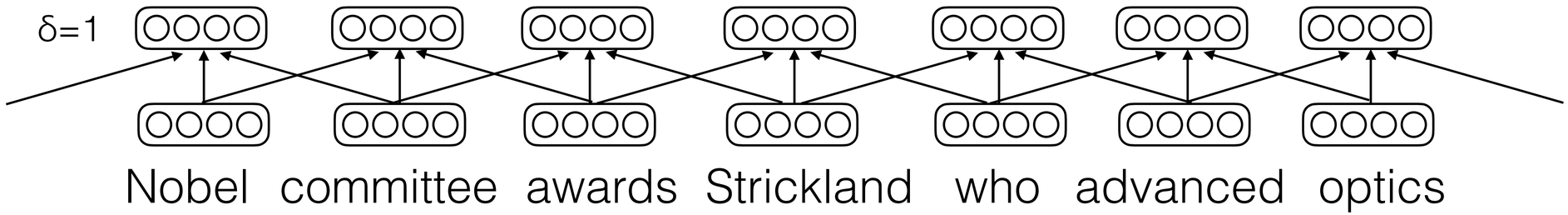
Sequence labeling w/ dilated CNNs

Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width δ**

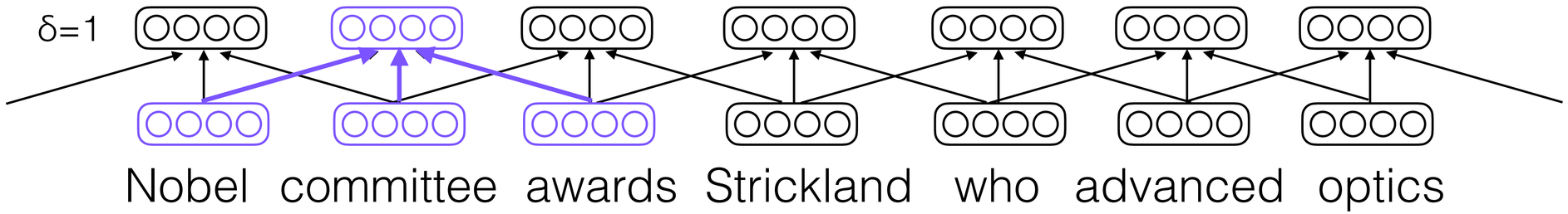
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



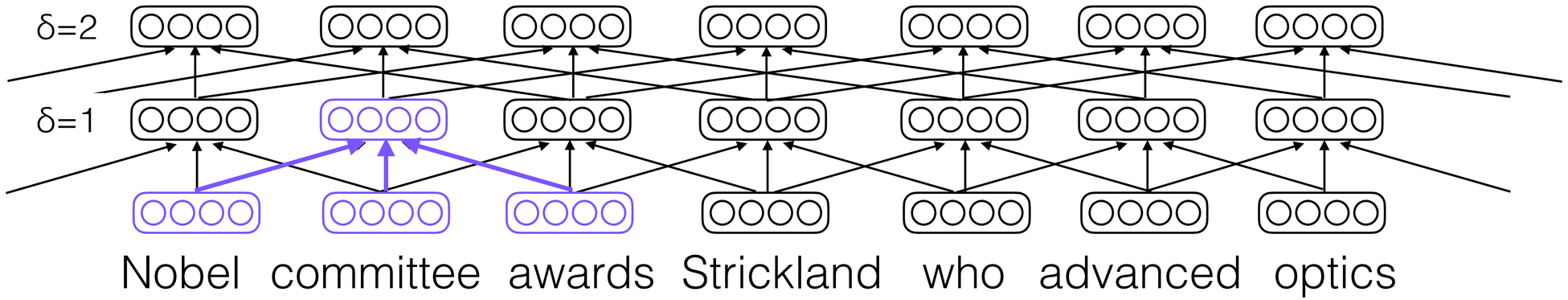
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



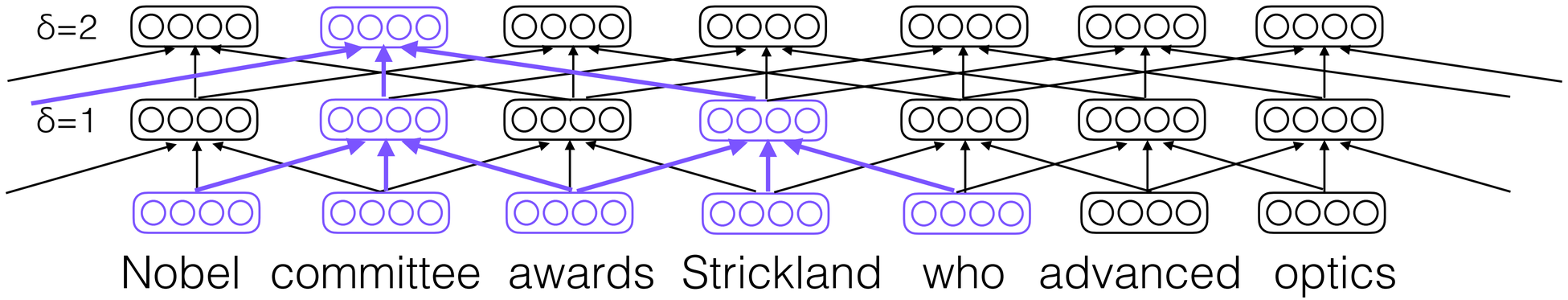
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



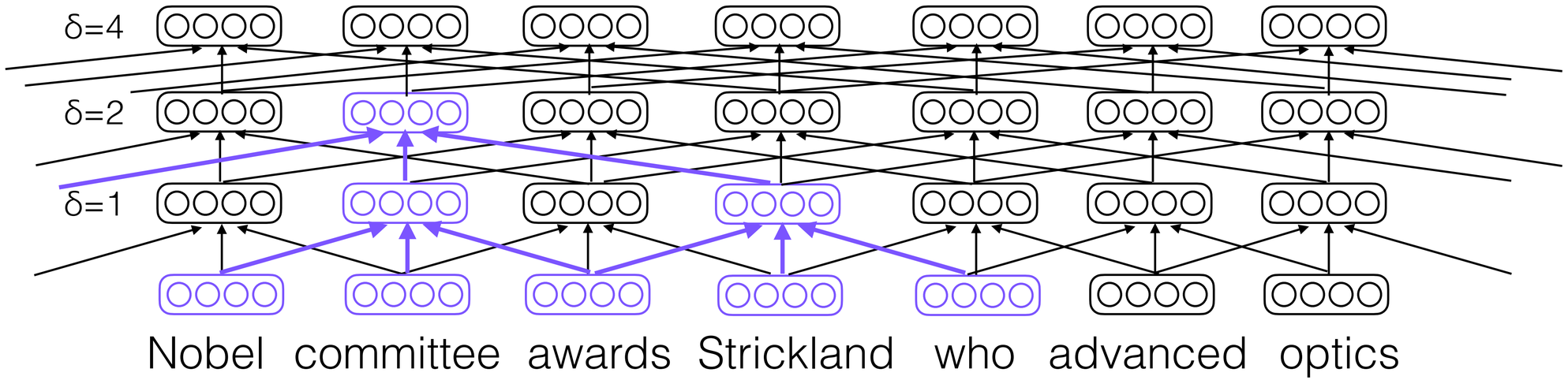
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



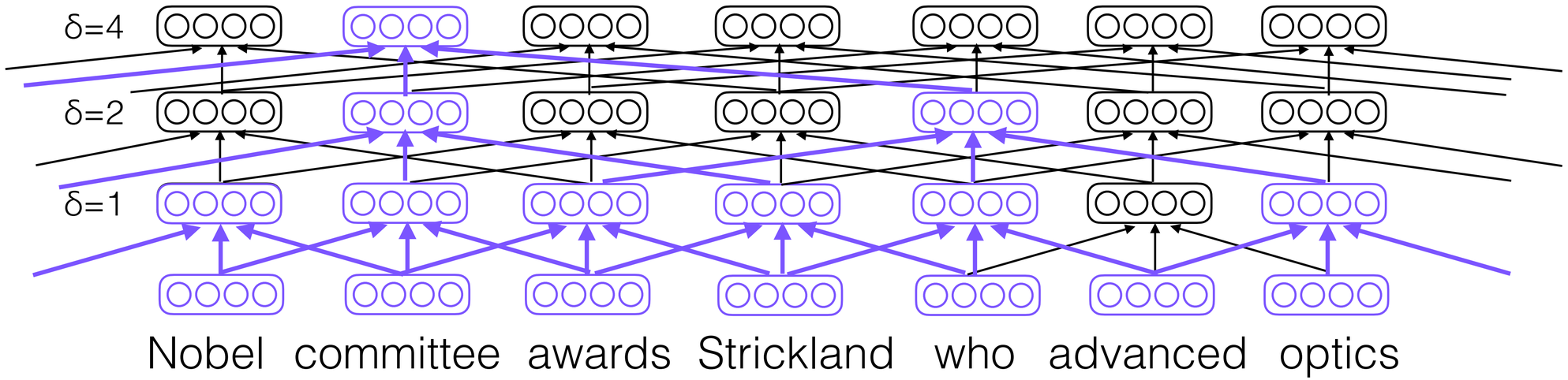
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



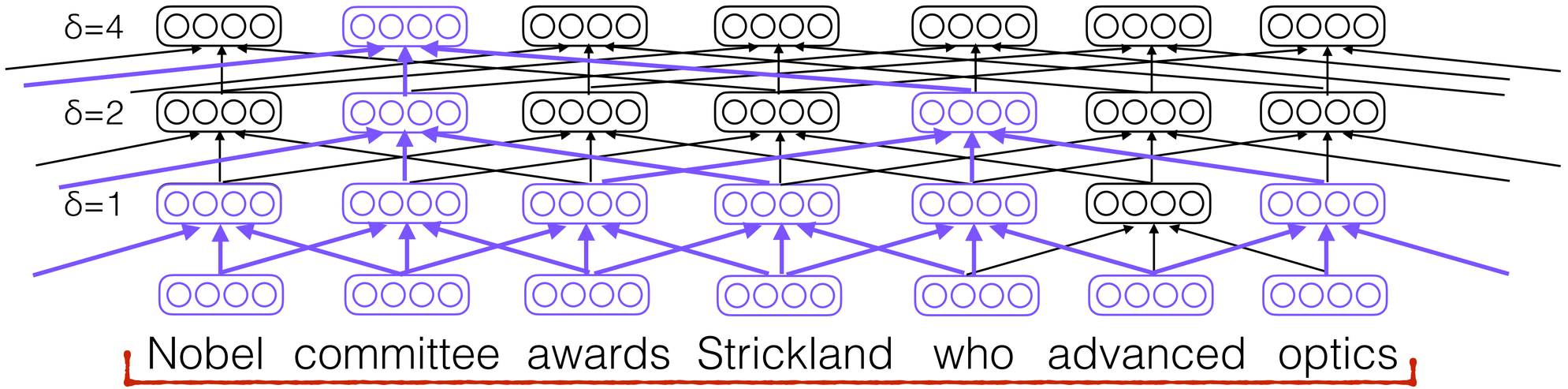
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



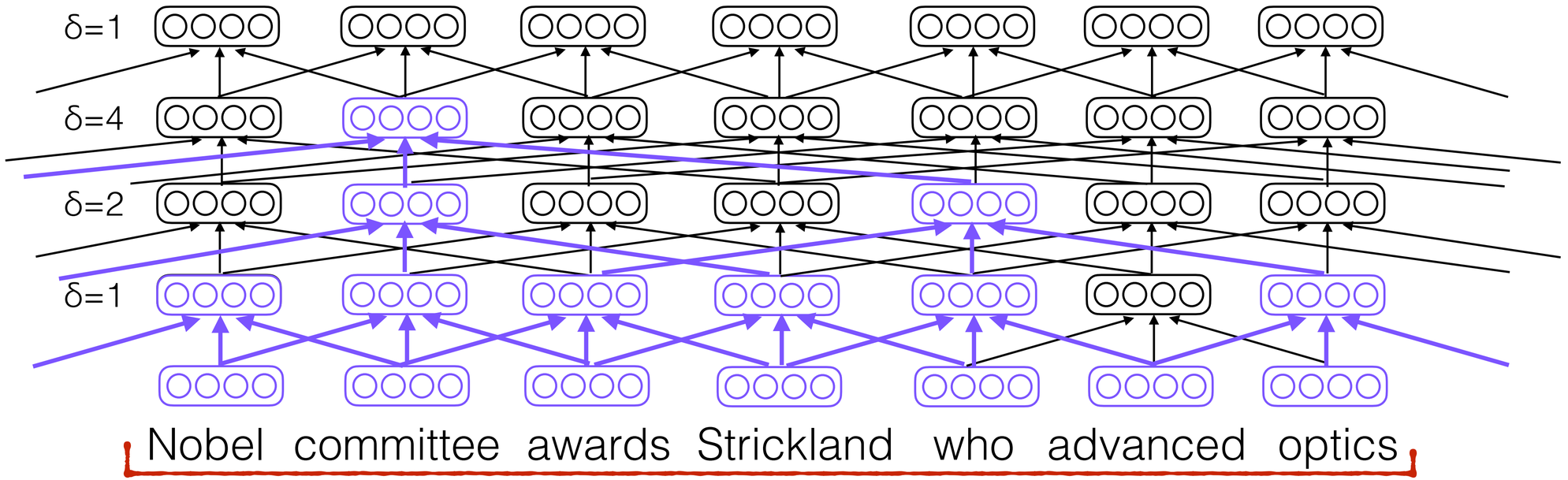
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



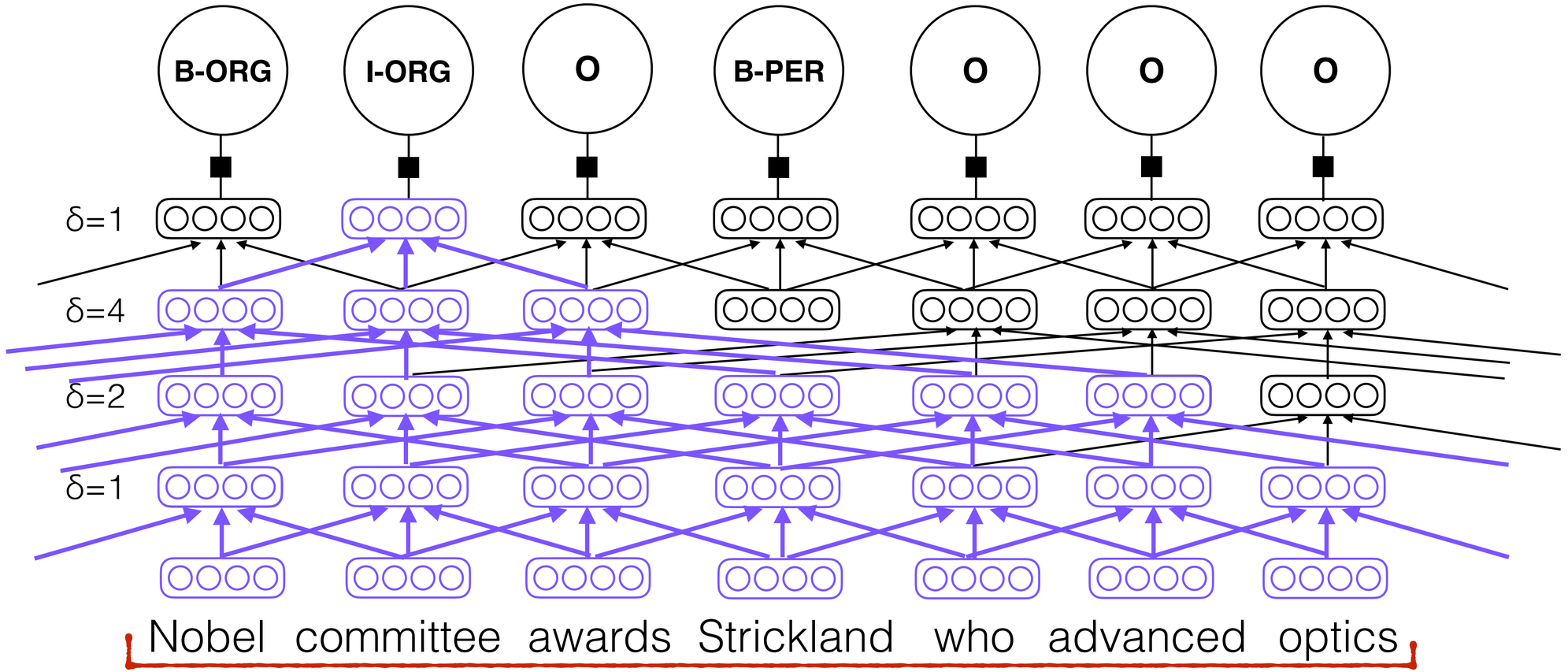
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



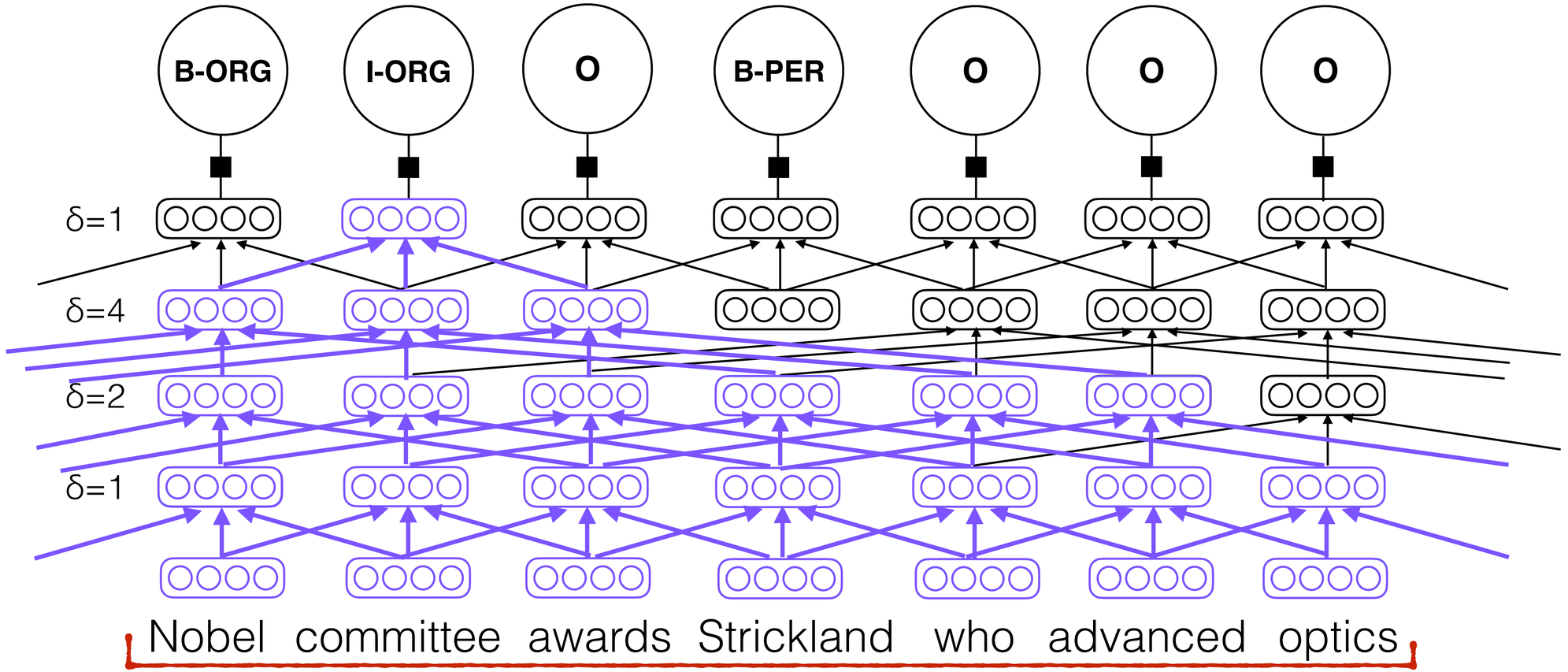
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***



Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width *δ***
- Context window grows **exponentially** w/ number of layers.



Sequence labeling w/ dilated CNNs

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

	NER F1	
	sentence	**document**
Bi-LSTM-CRF	90.4 ± 0.1	90.6 ± 0.2
ID-CNN (ours)	90.3 ± 0.3	90.7 ± 0.2

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

	NER F1	
	sentence	**document**
Bi-LSTM-CRF	90.4 ± 0.1	90.6 ± 0.2
ID-CNN (ours)	90.3 ± 0.3	90.7 ± 0.2

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

	NER F1	
	sentence	**document**
Bi-LSTM-CRF	90.4 ± 0.1	90.6 ± 0.2
ID-CNN (ours)	90.3 ± 0.3	90.7 ± 0.2

14x speed-up

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

	NER F1	
	sentence	**document**
Bi-LSTM-CRF	90.4 ± 0.1	90.6 ± 0.2
ID-CNN (ours)	90.3 ± 0.3	90.7 ± 0.2

14x speed-up

Sequence labeling w/ dilated CNNs

- Why use a (dilated) CNN over a (bidirectional) LSTM?
- Efficiency (on GPUs). Representations for every token in the sequence can be computed in parallel for CNN; linear dependence on sequence length for LSTM.

	NER F1	
	sentence	**document**
Bi-LSTM-CRF	90.4 ± 0.1	90.6 ± 0.2
ID-CNN (ours)	90.3 ± 0.3	90.7 ± 0.2
	14x speed-up	8x speed-up

Character embeddings

Character embeddings

- Character-level representations of words help to deal with UNKs.

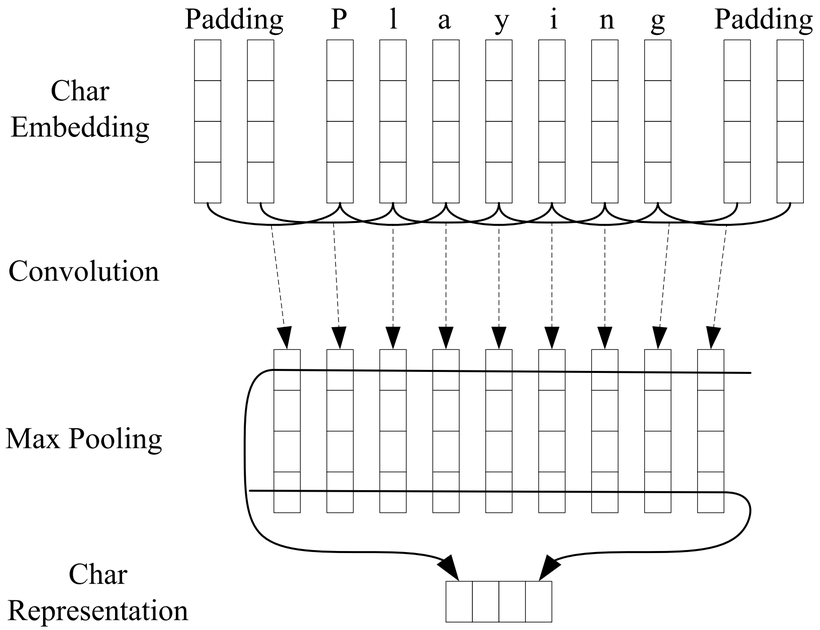
Character embeddings

- Character-level representations of words help to deal with UNKs.
- Usually, CNNs + pooling are used to compose characters into word embeddings.

Character embeddings

- Character-level representations of words help to deal with UNKs.
- Usually, CNNs + pooling are used to compose characters into word embeddings.

CNN + max pooling

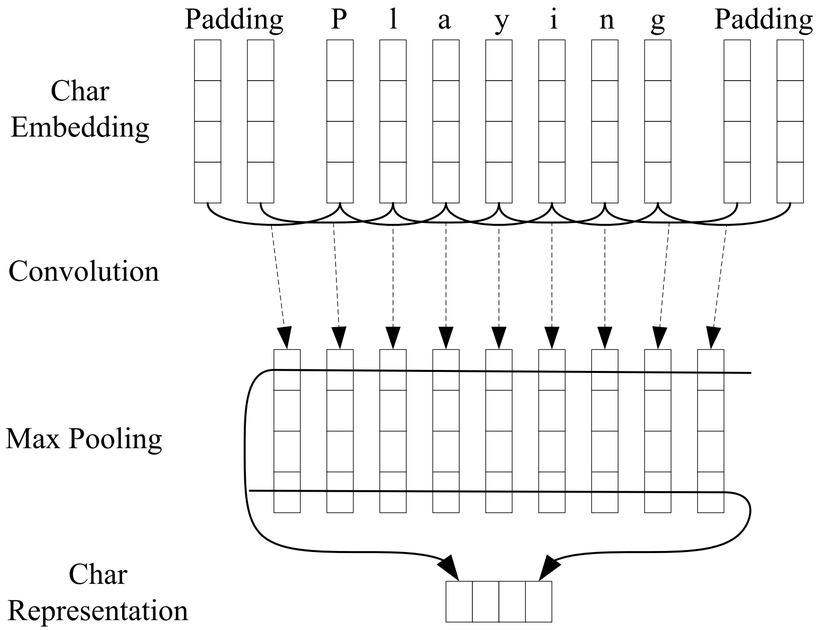


Images from: Ma and Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. ACL 2016.
Lample et al. Neural Architectures for Named Entity Recognition. NAACL 2016.

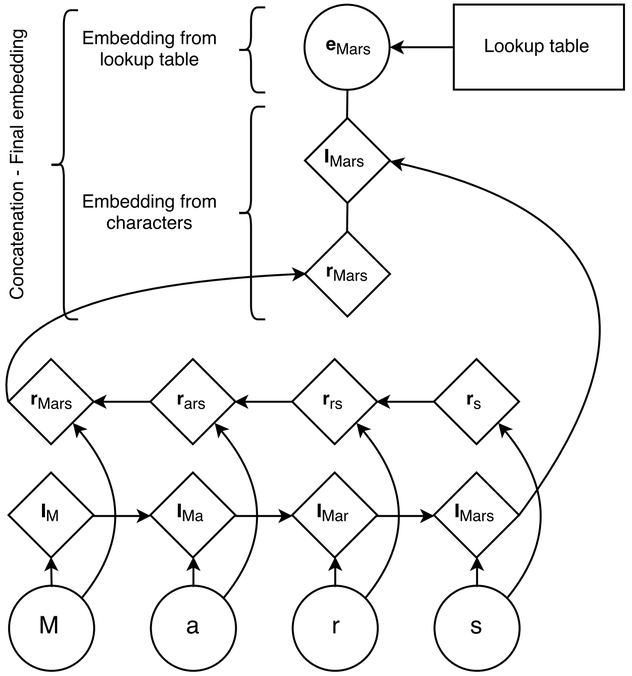
Character embeddings

- Character-level representations of words help to deal with UNKs.
- Usually, CNNs + pooling are used to compose characters into word embeddings.

CNN + max pooling



bidirectional LSTM

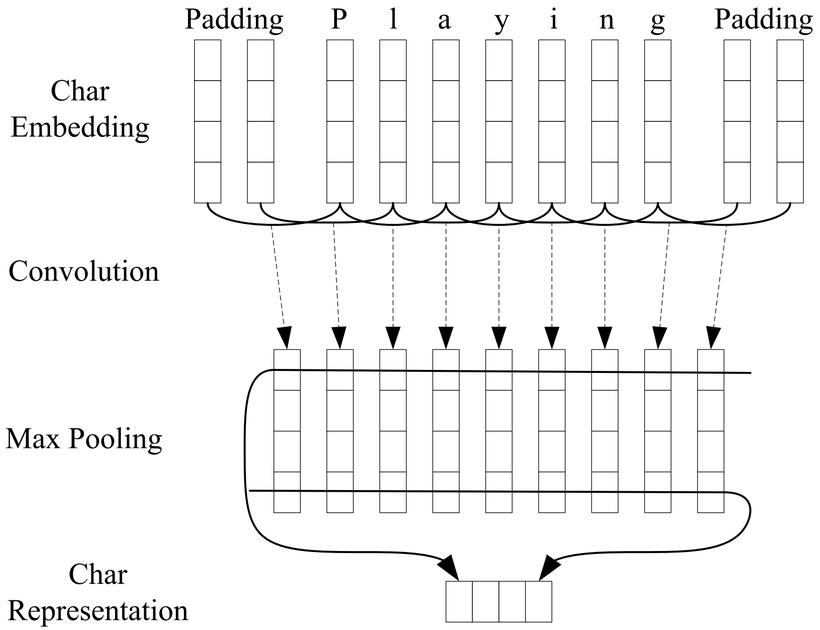


Images from: Ma and Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. ACL 2016.
Lample et al. Neural Architectures for Named Entity Recognition. NAACL 2016.

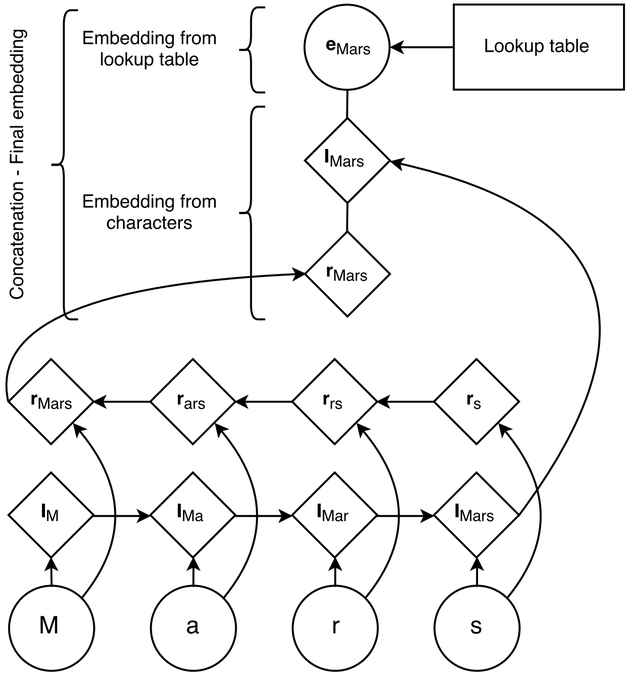
Character embeddings

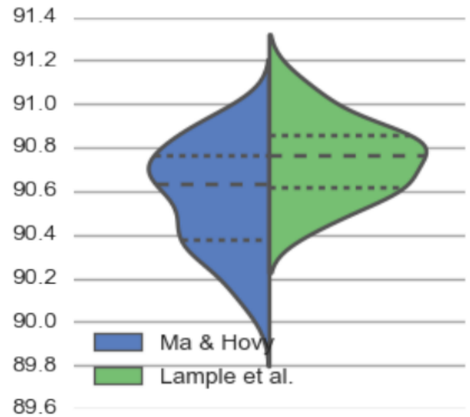
- Character-level representations of words help to deal with UNKs.
- Usually, CNNs + pooling are used to compose characters into word embeddings.

CNN + max pooling



bidirectional LSTM





Images from: Ma and Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. ACL 2016.
Lample et al. Neural Architectures for Named Entity Recognition. NAACL 2016.

Multilingual part-of-speech tagging

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

partilerindeydi

partisindeydiler

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

partilerindeydi

partisindeydiler

he/she/they(sing) were/was at their(plur) party

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

partilerindeydi

he/she/they(sing) were/was at their(plur) party

partisindeydiler

they(plur) were at his/her/their(sing) party

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

partilerindeydi

party

he/she/they(sing) were/was at their(plur) party

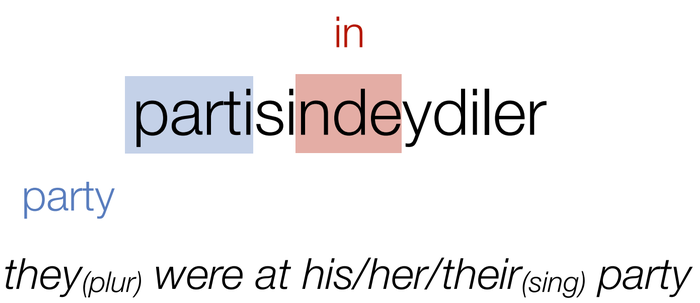
partisindeydiler

party

they(plur) were at his/her/their(sing) party

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology



Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

Multilingual part-of-speech tagging

- Many UNKs in morphologically-rich languages like Czech, Hungarian, Turkish
  - 250,000 word corpus of Hungarian has > 2x as many types as a similarly sized corpus of English
  - 10 million word corpus of Turkish contains 4x as many types
- Information coded in morphology

Yerdeki **izin** temizlenmesi gerek.
The trace on the floor should be cleaned.

iz + `Noun+A3sg+Pnon+Gen`

Multilingual part-of-speech tagging

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

姚明进入总决赛

Yao Ming reaches the finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

姚明进入总决赛

Yao Ming reaches the finals

姚明　进入　总决赛

YaoMing　reaches　finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

姚明进入总决赛

Yao Ming reaches the finals

CTB

姚明　进入　总决赛

YaoMing　reaches　finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

姚明进入总决赛

Yao Ming reaches the finals

CTB

姚明　进入　总决赛

YaoMing　reaches　finals

姚　明　进入　总　决赛

Yao　Ming　reaches　overall　finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.

姚明进入总决赛

Yao Ming reaches the finals

CTB

姚明 进入 总决赛

YaoMing reaches finals

Peking U.

姚 明 进入 总 决赛

Yao Ming reaches overall finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.
- UNKs are difficult: majority of unknown words are common nouns and verbs due to compounding

姚明进入总决赛

Yao Ming reaches the finals

CTB

姚明 进入 总决赛

YaoMing reaches finals

Peking U.

姚 明 进入 总 决赛

Yao Ming reaches overall finals

Multilingual part-of-speech tagging

- In non-word-space languages like Chinese, word segmentation is either applied before tagging or performed jointly.
- UNKs are difficult: majority of unknown words are common nouns and verbs due to compounding

姚明进入总决赛

Yao Ming reaches the finals

CTB

姚明 进入 总决赛

YaoMing reaches finals

Peking U.

姚 明 进入 总 决赛

Yao Ming reaches overall finals

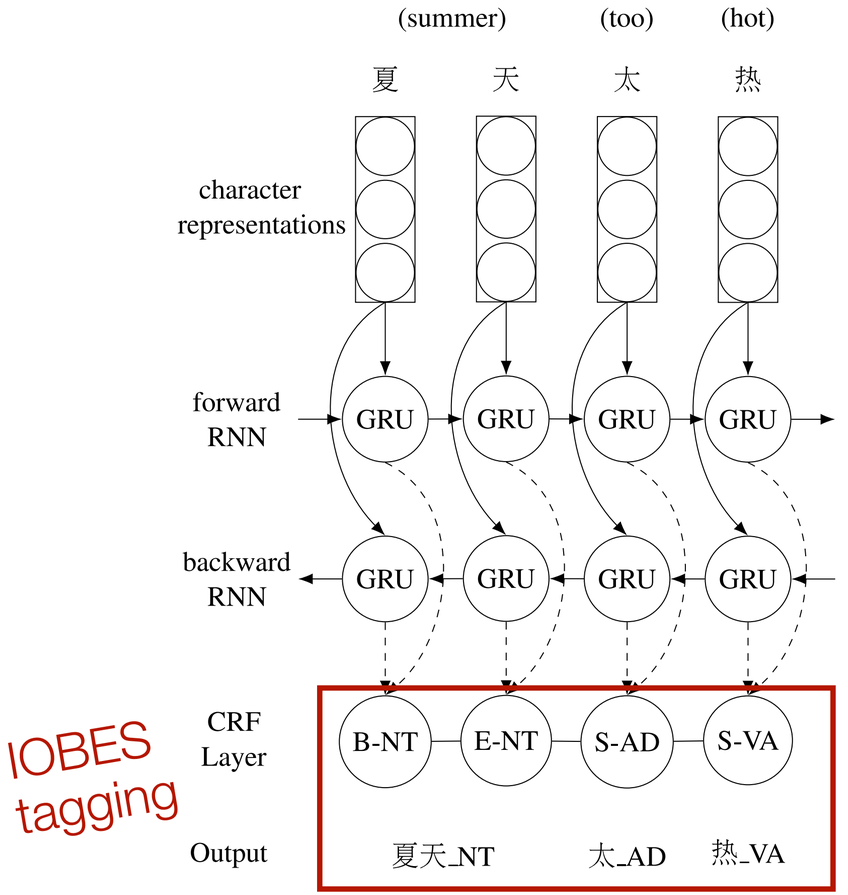


Figure from: Shao et al. Character-based Joint Segmentation and POS Tagging for Chinese using Bidirectional RNN-CRF. IJCNLP 2017.

Multilingual part-of-speech tagging

Multilingual part-of-speech tagging

- **Universal POS tags** [Petrov et al. 2012] provide a cross-lingual tag set.

Language	Source	# Tags
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	34
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	30
German	Tiger/CoNLL06 (Brants et al., 2002)	54
German	Negra (Skut et al., 1997)	54
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42
Korean	Sejong (http://www.sejong.or.kr)	187
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	22
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31

Table from: Petrov, Das and McDonald. A Universal Part-of-Speech Tagset. LREC 2012.

Multilingual part-of-speech tagging

- **Universal POS tags** [Petrov et al. 2012] provide a cross-lingual tag set.
  - Coarse grained: 16 tags

open class	closed class	other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Multilingual part-of-speech tagging

- **Universal POS tags** [Petrov et al. 2012] provide a cross-lingual tag set.
  - Coarse grained: 16 tags
  - Finer-grained analysis split off into morphological tags (case, gender, …)

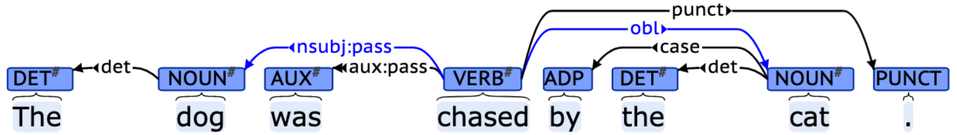
open class	closed class	other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Multilingual part-of-speech tagging

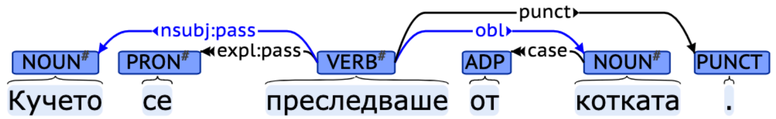
- **Universal POS tags** [Petrov et al. 2012] provide a cross-lingual tag set.
  - Coarse grained: 16 tags
  - Finer-grained analysis split off into morphological tags (case, gender, …)

open class	closed class	other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

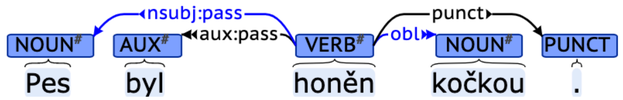
English:



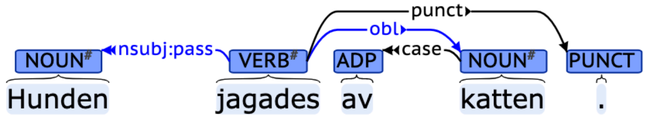
Bulgarian:



Czech:



Swedish:



Example from: https://universaldependencies.org/

Announcements

- **Project 2 released today after class**: sequence labeling.
  - Due: October 16.
  - You will implement part-of-speech taggers for English and Norwegian:
  - HMM, BiLSTM, and BiLSTM-CRF.
- Friday's recitation will be an overview of P2.