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!

\[
P(y \mid w) = \frac{\exp(\psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\psi(w, y'))}
\]
Conditional random fields (CRFs)

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

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

\(y'\) is an entire sequence
Conditional random fields (CRFs)

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

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

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

\(y'\) is an entire sequence
Conditional random fields (CRFs)

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

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

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

\(y'\) is an entire sequence
decompose into local scores
Conditional random fields (CRFs)

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

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

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

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

\(y'\) is an entire sequence
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]
**Conditional random fields (CRFs)**

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

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

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>...</th>
<th>1</th>
<th>...</th>
<th>0</th>
<th>...</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>...</th>
<th>0</th>
<th>VB?</th>
</tr>
</thead>
</table>
Conditional random fields (CRFs)

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

\[ \psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m) \]

1 0 ... 1 ... 0 ... 1 0 0 0 0 0 ... 0

Janet will back the bill

\( f(w, \text{VB}, \text{MD}, 3) \)
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]

\[
f(w, \text{VB, MD, 3}) = f(w_1, w_2, w_3, w_4, \text{VB, MD})
\]
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]

\[
f(w, VB, MD, 3) = f(w_1, w_2, w_3, w_4, VB, MD)
\]
Conditional random fields (CRFs)

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

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

$$f(w, \text{VB}, \text{MD}, 3) = f(w_1, w_2, w_3, w_4, \text{VB}, \text{MD})$$
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]

\[
f(w, VB, MD, 3) = f(w_1, w_2, w_3, w_4, VB, MD)
\]
Conditional random fields (CRFs)

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

\[
\psi(w, y) = \sum_{m=1}^{M+1} \theta \cdot f(w, y_m, y_{m-1}, m)
\]

\[
f(w, \text{VB, MD, 3}) = f(w_1, w_2, w_3, w_4, \text{VB, MD})
\]
Conditional random fields (CRFs)

- Decoding:

\[ P(y \mid w) = \frac{\exp(\Psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\Psi(w, y'))} \]
Conditional random fields (CRFs)

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

\[
P(y \mid w) = \frac{\exp(\Psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\Psi(w, y'))}
\]
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

\[ \hat{y} = \arg\max_y \log P(y \mid w) \]

\[ P(y \mid w) = \frac{\exp(\Psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\Psi(w, y'))} \]
**Conditional random fields (CRFs)**

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

\[
\hat{y} = \arg\max_y \log P(y \mid w) = \arg\max_y \Psi(y, w) - \log \sum_{y' \in \mathcal{Y}(w)} \exp \Psi(y', w)
\]

\[
P(y \mid w) = \frac{\exp(\Psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\Psi(w, y'))}
\]
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

\[
\hat{y} = \arg\max_y \log P(y | w) = \arg\max_y \psi(y, w) - \log \sum_{y' \in \mathcal{Y}(w)} \exp \psi(y', w)
\]

\[
P(y | w) = \frac{\exp(\psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\psi(w, y'))}
\]
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

\[
\hat{y} = \arg\max_y \log P(y | w) = \arg\max_y \Psi(y, w) - \log \sum_{y' \in \mathcal{Y}(w)} \exp \Psi(y', w)
\]

\[
P(y | w) = \frac{\exp(\Psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\Psi(w, y'))}
\]

same for all settings of \(y\)
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

$$\hat{y} = \arg \max_y \log P(y | w)$$

$$= \arg \max_y \psi(y, w) - \log \sum_{y' \in \mathcal{Y}(w)} \exp \psi(y', w)$$

$$= \arg \max_y \psi(y, w) \quad \text{same for all settings of } y$$

$$= \arg \max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m)$$

$$P(y | w) = \frac{\exp(\psi(w, y))}{\sum_{y' \in \mathcal{Y}(w)} \exp(\psi(w, y'))}$$
Conditional random fields (CRFs)

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

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

\[
\hat{y} = \arg \max_y \log P(y \mid w)
\]

\[
= \arg \max_y \Psi(y, w) - \log \sum_{y' \in \mathcal{Y}(w)} \exp \Psi(y', w)
\]

\[
= \arg \max_y \Psi(y, w) \quad \text{same for all settings of } y
\]

\[
= \arg \max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\]
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

\[
\hat{y} = \arg\max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\]
Conditional random fields (CRFs)

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

\[
\hat{y} = \arg\max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\]

\[
\max_{y_{1:M}} \Psi(w, y_{1:M}) = \max_{y_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})
\]
Conditional random fields (CRFs)

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

\[ \hat{y} = \text{argmax}_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1}) \]

score of best tag sequence of length M

\[ \max_{y_1:M} \Psi(w, y_1:M) = \max_{y_1:M} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1}) \]
Conditional random fields (CRFs)

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

\[
\hat{y} = \arg\max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\]

score of best tag sequence of length M

\[
\max_{y_1:M} \Psi(w, y_1:M) = \max_{y_1:M} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})
\]

\[
= \max_{y_M} s_{M+1}(\langle s \rangle, y_M) + \max_{y_1:M-1} \sum_{m=1}^{M} s_m(y_m, y_{m-1})
\]
Conditional random fields (CRFs)

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

\[ \hat{y} = \arg\max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1}) \]

score of best tag sequence of length M

\[ \max_{y_{1:M}} \Psi(w, y_{1:M}) = \max_{y_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1}) \]

\[ = \max_{y_M} s_{M+1}(\langle \psi \rangle, y_M) + \max_{y_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1}) \]
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

$$\hat{y} = \arg\max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})$$

- **Score of best tag sequence of length** $M$
  $$\max_{y_{1:M}} \Psi(w, y_{1:M}) = \max_{y_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})$$

- **Score of best tag sequence of length** $M-1$
  $$\max_{y_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})$$

- **Score of most probable extension** $y_M$
  $$\max_{y_M} s_{M+1}(\langle s \rangle, y_M)$$
Conditional random fields (CRFs)

**Decoding:** Direct application of Viterbi!

\[ \hat{y} = \arg \max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1}) \]

- **Score of best tag sequence of length M**
  \[ \max_{y_{1:M}} \psi(w, y_{1:M}) = \max_{y_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1}) \]
  \[ = \max_{y_M} s_{M+1}(\langle/s\rangle, y_M) + \max_{y_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1}) \]

- **Score of most probable extension** \( y_M \)

- **Same subproblem!**
**Conditional random fields (CRFs)**

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

\[
\hat{y} = \arg \max_y \sum_{m=1}^{M+1} \psi(w, y_m, y_{m-1}, m) = s_m(y_m, y_{m-1})
\]

\[
\max \Psi(w, y_{1:M}) = \max_{y_{1:M}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1}) = \max_{y_{M}} s_{M+1}(\langle /s \rangle, y_M) + \max_{y_{1:M-1}} \sum_{m=1}^{M} s_m(y_m, y_{m-1})
\]
Learning in CRFs
Learning in CRFs

As with logistic regression, weights $\theta$ are learned by minimizing negative log likelihood:

$$
\ell = - \sum_{i=1}^{N} \log P(y^{(i)} | w^{(i)}; \theta)
$$
Learning in CRFs

As with logistic regression, weights $\theta$ are learned by minimizing negative log likelihood:

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

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

As with logistic regression, weights $\theta$ are learned by minimizing negative log likelihood:

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

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

sum over all possible labelings
Learning in CRFs

- As with logistic regression, weights $\theta$ are learned by minimizing negative log likelihood:

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

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

- 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:

\[\alpha_m(y_m) = \sum_{y_1:m-1} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1})\]
Learning in CRFs

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

\[
\alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})
\]
Learning in CRFs

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

\[ \alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1}) \]

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

\[ = \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{y_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1}) \]
Learning in CRFs

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

\[ \alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \left( \sum_{n=1}^{m} s_n(y_n, y_{n-1}) \right) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1}) \]

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

\[
= \sum_{y_{m-1}} \left( \exp s_m(y_m, y_{m-1}) \right) \sum_{y_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1}) \\
= \sum_{y_{m-1}} \left( \exp s_m(y_m, y_{m-1}) \right) \times \alpha_{m-1}(y_{m-1})
\]
Learning in CRFs

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

\[
\alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})
\]

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

\[
\begin{align*}
= & \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{y_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1}) \\
= & \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})
\end{align*}
\]

sum instead of max
Learning in CRFs

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

\[
\alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \sum_{n=1}^{m} s_n(y_n, y_{n-1}) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1})
\]

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

\[
= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{y_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})
\]

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

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:

\[ \alpha_m(y_m) = \sum_{y_{1:m-1}} \exp \left( \sum_{n=1}^{m} s_n(y_n, y_{n-1}) \right) = \sum_{y_{1:m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1}) \]

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

\[
= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{y_{1:m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1})
= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})
\]

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:

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

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

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

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

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

- 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:

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

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:

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

- 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:

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

- 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:

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

- Gradients can be computed by automatic differentiation!

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

\[
\alpha_{m-1}(k') \quad \exp s_m(k, k') \quad \beta_m(k)
\]

**forward score:** sum over all prefixes

**backward score:** sum over all suffixes

The result is product of three terms: a score that sums over all the ways to get to the suffixes \(Y_0 = n\), a score for the transition from \(Y_k = Y_{k+1} = Y_i\) for each \(k\) from \(0\) to \(n-1\), and a score that sums over ways to finish the sequence from \(Y_{n-1} = Y_n\). The first term of Equation 7.87 is equal to the forward score, \(\alpha_m = \sum_{k'} \alpha_{m-1}(k') \exp s_m(k, k') \beta_m(k)\). The second term is the backward score, \(\sum_{Y_{n-1} = k'} \mathbb{E}[f_j(w^{(i)}, y)] \). The third term — the sum over ways to finish the sequence from \(Y_{m-1} = k'\), beginning with the tag sequence \(y^{(i)}\) for token sequence \(w^{(i)}\), is also known as the **sequence margin** of the forward-backward algorithm.
Learning in CRFs

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

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

- Gradients can be computed by automatic differentiation!

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

To understand this computation, compare with the forward recurrence in Equation 7.81.
Better features for sequence labeling?

- Until now: hand-engineered features:
Better features for sequence labeling?

- Until now: hand-engineered features:

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>( f_{\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}))</td>
</tr>
<tr>
<td>POS</td>
<td>( p_{i-{3,2,1}}, a_{i-{0,1,2,3}}, (p_{i-2,i-1}), (a_{i+1,i+2}))</td>
</tr>
<tr>
<td>Affix</td>
<td>( c_1, c_2, c_3, c_n, c_{n-1}, c_{n-2}, c_{n-3})</td>
</tr>
<tr>
<td>Binary</td>
<td>initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)</td>
</tr>
</tbody>
</table>

- \( w_i \) contains a particular prefix (from all prefixes of length \( \leq 4 \))
- \( w_i \) contains a particular suffix (from all suffixes of length \( \leq 4 \))
- \( w_i \) 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:

<table>
<thead>
<tr>
<th>Lexical</th>
<th>( f_{\pm(0,1,2,3), (m_{i-2}, i-1), (m_{i-1}, i), (m_{i-1}, i+1), (m_{i+1}, i), (m_{i+1}, i+2), (m_{i+1}, i+3)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS</td>
<td>( 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}) )</td>
</tr>
<tr>
<td>Affix</td>
<td>( c_1, c_2, c_3, c_n, c_{n-1}, c_{n-2}, c_{n-3} )</td>
</tr>
<tr>
<td>Binary</td>
<td>initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)</td>
</tr>
</tbody>
</table>

\[ f(w, y_m, y_{m-1}, m) = \begin{bmatrix} 1 & 0 & \ldots & 1 & \ldots & 0 & \ldots & 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \]

\( w_i \) contains a particular prefix (from all prefixes of length \( \leq 4 \))
\( w_i \) contains a particular suffix (from all suffixes of length \( \leq 4 \))
\( w_i \) 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.

Janet will back the bill.
Better features for sequence labeling?

- Until now: hand-engineered features:

  **pros:**

  **cons:**

$$f(w, y_m, y_{m-1}, m) = \begin{array}{cccccccccccc}
1 & 0 & \ldots & 1 & \ldots & 0 & \ldots & 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
\text{w}_{m-1} = \text{will} & \text{w}_m = \text{my} & \text{w}_{m+1} = \text{ache} & y_{m-1} = \text{MD} & \text{Janet will back the bill}.
\end{array}$$
Better features for sequence labeling?

- Until now: hand-engineered features:

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

  $f(w, y_m, y_{m-1}, m) = \begin{bmatrix}
  1 & 0 & \cdots & 1 & \cdots & 0 & \cdots & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
  \end{bmatrix}$

  $W_{m-1} = \text{will}$  $W_m = \text{my}$  $W_{m+1} = \text{ache}$  $y_{m-1} = \text{MD}$  Janet will back the bill.
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

\[
f(w, y_m, y_{m-1}, m) = \begin{pmatrix}
1 & 0 & \ldots & 1 & \ldots & 0 & \ldots & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0
\end{pmatrix}
\]

Janet will back the bill.
Neural sequence labeling

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

$$f(w, y_m, y_{m-1}, m) =$$
Neural sequence labeling

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

**pros:**

**cons:**

$$f(w, y_m, y_{m-1}, m) = \cdots$$
Neural sequence labeling

Parameterize $f$ with a (deep) neural network.

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

**cons:**

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

![Diagram of neural network]

Pros:

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

$$f(w, y_m, y_{m-1}, m) = \ldots$$
Neural sequence labeling
Neural sequence labeling

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

word embeddings
Neural sequence labeling

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

word embeddings

neural network
Neural sequence labeling

<\s> Janet will back the bill \</s>
Neural sequence labeling

per-token features

word embeddings

<s> Janet will back the bill </s>
Neural sequence labeling
Bidirectional RNNs

word embeddings

<s> Janet will back the bill </s>
Neural sequence labeling

Bidirectional RNNs
Neural sequence labeling
Bidirectional RNNs

<s> Janet will back the bill </s>
Neural sequence labeling

Bidirectional RNNs

per-token features

forward RNN

backward RNN

word embeddings

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

concatenate
Neural sequence labeling
Bidirectional RNNs

softmax( )

concatenate

per-token features

forward RNN
backward RNN
word embeddings

<s> Janet will back the bill </s>
Neural sequence labeling
Bidirectional RNNs

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

per-token features
forward RNN
backward RNN
word embeddings
Neural sequence labeling
Bidirectional RNNs

Word embeddings

Forward RNN

Backward RNN

Per-token features

<s> Janet will back the bill </s>
Neural sequence labeling
Bidirectional RNNs

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

Word embeddings
Forward RNN
Backward RNN
Per-token features

NNP MD VB? DT? NN?
Neural sequence labeling
Bidirectional RNNs

per-token features
forward RNN
backward RNN
word embeddings

<s> Janet will back the bill </s>
Neural sequence labeling
Bidirectional RNNs

Word embeddings

Forward RNN

Backward RNN

Per-token features

<s> Janet will back the bill </s>
Neural sequence labeling

Bidirectional RNNs

![Diagram showing a neural network model for sequence labeling using bidirectional RNNs.](image-url)

- **Word Embeddings**
- **Forward RNN**
- **Backward RNN**
- **Per-token Features**

The diagram illustrates the flow of information through a bidirectional RNN network, with input words and corresponding part-of-speech tags. The network processes each word in both forward and backward directions, integrating per-token features to label each word.

Input: `<s>` Janet will back the bill `</s>`

Output: NNP MD VB? DT? NN?
Neural sequence labeling
Bidirectional RNN-CRFs

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

word embeddings
forward RNN
backward RNN
per-token features
Neural sequence labeling

Neural network

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

per-token features

word embeddings
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.
**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).

![Diagram](image.png)

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.

```
<s> Janet  will  back  the  bill  [d_word]
```

 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.

- The dimensions of the convolutional layer are calculated as follows:
  - $([k_d]d_{word})$ for the kernel size $k_d$.
  - $[d_{word}]$ for the word dimension $d_{word}$.

Example:

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

Kernel size = 3
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.

```
dims:
[d_z]
[kd_word x d_z]
[kd_word]
[d_word]
```
In NLP, CNNs merge information across contiguous, fixed-width spans of tokens.

**dims:**
- \([d_z]\)
- \([kd_{\text{word}} \times d_z]\)
- \([kd_{\text{word}}]\)
- \([d_{\text{word}}]\)
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.

 dims: $[d_z \times \text{# filters}]$
 $[kd_{\text{word}} \times d_z]$
 $[kd_{\text{word}}]$
 $[d_{\text{word}}]$
Sequence labeling w/ CNNs

B-ORG  I-ORG  O  B-PER  O  O  O

Nobel  committee  awards  Strickland  who  advanced  optics
Sequence labeling w/ CNNs

B-ORG  I-ORG  O  B-PER  O  O  O

Nobel  committee  awards  Strickland  who  advanced  optics
Sequence labeling w/ CNNs

B-ORG  I-ORG  O  B-PER  O  O  O

Nobel  committee  awards  Strickland  who  advanced  optics
Sequence labeling w/ CNNs

B-ORG  I-ORG  O  B-PER  O  O  O

Nobel  committee  awards  Strickland  who  advanced  optics
Sequence labeling w/ CNNs

B-ORG  I-ORG  O  B-PER  O  O  O

Nobel  committee  awards  Strickland  who  advanced  optics
Sequence labeling w/ CNNs

Nobel committee awards Strickland who advanced optics
Sequence labeling w/ CNNs

Nobel committee awards Strickland who advanced optics
Sequence labeling w/ CNNs

Nobel committee awards Strickland who advanced optics

encode in parallel
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].

Nobel committee awards Strickland who advanced optics
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* $\delta$
Sequence labeling w/ dilated CNNs

- Additional parameter: *dilation width* $\delta$

$\delta=1$

Nobel committee awards Strickland who advanced optics
Sequence labeling w/ dilated CNNs

- Additional parameter: dilation width $\delta$
Sequence labeling w/ dilated CNNs

- Additional parameter: dilation width $\delta$
Sequence labeling w/ dilated CNNs

- Additional parameter: \textit{dilation width} $\delta$

\[ \delta = 1 \]

\[ \delta = 2 \]
Sequence labeling w/ dilated CNNs

- Additional parameter: dilation width $\delta$

$\delta=1$

$\delta=2$

$\delta=4$

Nobel committee awards Strickland who advanced optics
Sequence labeling w/ dilated CNNs

- Additional parameter: dilation width $\delta$
Sequence labeling w/ dilated CNNs

- Additional parameter: *dilation width $\delta$*

\[
\begin{align*}
\delta &= 4 \\
\delta &= 2 \\
\delta &= 1 \\
&\text{Nobel committee awards Strickland who advanced optics}
\end{align*}
\]
Sequence labeling w/ dilated CNNs

- Additional parameter: dilation width $\delta$
Sequence labeling w/ dilated CNNs

- Additional parameter: **dilation width** $\delta$
**Sequence labeling w/ dilated CNNs**

- Additional parameter: **dilation width** $\delta$

- Context window grows **exponentially** w/ number of layers.

---

$\delta = 1$

$\delta = 2$

$\delta = 4$

Nobel committee awards Strickland who advanced optics
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.

<table>
<thead>
<tr>
<th>NER F1</th>
<th>sentence</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM-CRF</td>
<td>90.4 ± 0.1</td>
<td>90.6 ± 0.2</td>
</tr>
<tr>
<td>ID-CNN (ours)</td>
<td>90.3 ± 0.3</td>
<td>90.7 ± 0.2</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th></th>
<th>NER F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sentence</td>
</tr>
<tr>
<td>Bi-LSTM-CRF</td>
<td>90.4 ± 0.1</td>
</tr>
<tr>
<td>ID-CNN (ours)</td>
<td>90.3 ± 0.3</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>sentence</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM-CRF</td>
<td>90.4 ± 0.1</td>
<td>90.6 ± 0.2</td>
</tr>
<tr>
<td>ID-CNN (ours)</td>
<td>90.3 ± 0.3</td>
<td>90.7 ± 0.2</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>sentence</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM-CRF</td>
<td>90.4 ± 0.1</td>
<td>90.6 ± 0.2</td>
</tr>
<tr>
<td>ID-CNN (ours)</td>
<td>90.3 ± 0.3</td>
<td>90.7 ± 0.2</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>NER F1</th>
<th>sentence</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-LSTM-CRF</td>
<td>90.4 ± 0.1</td>
<td>90.6 ± 0.2</td>
</tr>
<tr>
<td>ID-CNN (ours)</td>
<td>90.3 ± 0.3</td>
<td>90.7 ± 0.2</td>
</tr>
</tbody>
</table>

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

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**

Char Embedding → Convolution → Max Pooling → Char Representation

**bidirectional LSTM**

Embedding from lookup table → Embedding from characters → Lookup table

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

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    partisindeydidiler
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

partilerin-deydi  partisin-deydiler

he/she/they\textsubscript{(sing)} were/was at their\textsubscript{(plur)} party  they\textsubscript{(plur)} were at his/her/their\textsubscript{(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

He/she/they (sing) were/was at their (plur) 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

```
partilerindeydi
party their
he/she/they\text{(sing)} were/was at their\text{(plur)} party
```

```
partisindeydiler
party his/her/their\text{(sing)}
they\text{(plur)} were at his/her/their\text{(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

<table>
<thead>
<tr>
<th>Turkish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>partilerindeydi</td>
<td>in she/he/they (sing) was</td>
</tr>
<tr>
<td>partişindeydiler</td>
<td>in they (plur) were</td>
</tr>
<tr>
<td>he/she/they (sing) were/was at their (plur) party</td>
<td></td>
</tr>
<tr>
<td>they (plur) were at his/her/their (sing) party</td>
<td></td>
</tr>
</tbody>
</table>

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

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.

Yao Ming reaches the finals

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

姚明 进入 总决赛
YaoMing reaches finals

CTB

Peking U.
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.

**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.

- UNKs are difficult: majority of unknown words are common nouns and verbs due to compounding.

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.

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

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

<table>
<thead>
<tr>
<th>open class</th>
<th>closed class</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADP</td>
<td>PUNCT</td>
</tr>
<tr>
<td>ADV</td>
<td>AUX</td>
<td>SYM</td>
</tr>
<tr>
<td>INTJ</td>
<td>CCONJ</td>
<td>X</td>
</tr>
<tr>
<td>NOUN</td>
<td>DET</td>
<td></td>
</tr>
<tr>
<td>PROPN</td>
<td>NUM</td>
<td></td>
</tr>
<tr>
<td>VERB</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCONJ</td>
<td></td>
</tr>
</tbody>
</table>
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, …)

<table>
<thead>
<tr>
<th>open class</th>
<th>closed class</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADP</td>
<td>PUNCT</td>
</tr>
<tr>
<td>ADV</td>
<td>AUX</td>
<td>SYM</td>
</tr>
<tr>
<td>INTJ</td>
<td>CCONJ</td>
<td>X</td>
</tr>
<tr>
<td>NOUN</td>
<td>DET</td>
<td></td>
</tr>
<tr>
<td>PROPN</td>
<td>NUM</td>
<td></td>
</tr>
<tr>
<td>VERB</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCONJ</td>
<td></td>
</tr>
</tbody>
</table>
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, …)

<table>
<thead>
<tr>
<th>open class</th>
<th>closed class</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADP</td>
<td>PUNCT</td>
</tr>
<tr>
<td>ADV</td>
<td>AUX</td>
<td>SYM</td>
</tr>
<tr>
<td>INTJ</td>
<td>CCONJ</td>
<td>X</td>
</tr>
<tr>
<td>NOUN</td>
<td>DET</td>
<td></td>
</tr>
<tr>
<td>PROPN</td>
<td>NUM</td>
<td></td>
</tr>
<tr>
<td>VERB</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td>PRON</td>
<td>SCONJ</td>
<td></td>
</tr>
</tbody>
</table>

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.