

Natural Language Processing

Neural sequence labeling

Sachin Kumar

sachink@cs.cmu.edu

Includes slides/images from Graham Neubig, Emma Strubell, Wei Xu, Greg Durrett

Recap: Sequence Labeling

Input $\mathbf{x} = (x_1, \dots, x_n)$ Output $\mathbf{y} = (y_1, \dots, y_n)$

Part-of-Speech:

1 Lt. Gen. H.R. McMaster, a widely respected military strategist, is seen as one of the Army's leading intellectuals.

NP NP NP NP , DT RB JJ JJ NN , VBZ VBN IN CD IN DT NNP POS VBG
NNS .
 Lt. Gen. H.R. McMaster, a widely respected military strategist, is seen as one of the Army's leading intellectuals.

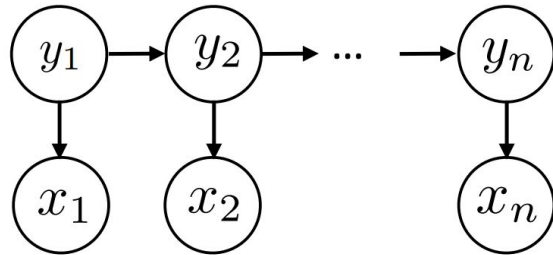
Named Entity Recognition:

1 Lt. Gen. H.R. McMaster, a widely respected military strategist, is seen as one of the Army's leading intellectuals.

Person Num Org
 Lt. Gen. H.R. McMaster, a widely respected military strategist, is seen as one of the Army's leading intellectuals.

○ ○ B-PER I-PER ○ ○ ○ ○

Recap: Generative Models of Sequence Labeling



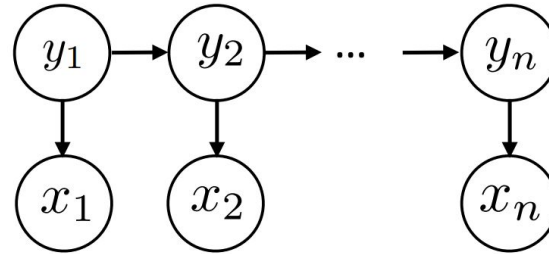
$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^n P(y_i | y_{i-1}) \prod_{i=1}^n P(x_i | y_i)$$

- Training: Maximum Likelihood Estimation (Count and Divide)

- Inference (decoding): $\operatorname{argmax}_{\mathbf{y}} P(\mathbf{y} | \mathbf{x}) = \operatorname{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$

- Viterbi: $\operatorname{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \operatorname{score}_{i-1}(y_{i-1})$

Limitations of HMMs



I will **friend** him on **Facebook**.

- Difficult to handle a word with an **unseen tag** or **an unknown word**
- HMMs don't allow adding additional features.
 - e.g. Facebook is capitalized (should be a NN), friend is followed by will, might be a VB.
- HMMs perform poorly on more complex tasks like Named Entity Recognition (NER)

Named Entity Recognition

B-PER I-PER O O O B-LOC O O O B-ORG O O
Barack Obama will travel to Hangzhou today for the G20 meeting .
 PERSON LOC ORG

- Why might an HMM not do so well here?
 - Lots of O's, so tags are not as informative [about the context].
 - Lots of unknown entities at test time – smoothing will not work.

Today: Discriminative Models

Goal: To tag a sequence

- Directly model $p(y|x)$
 - Simply using logistic regression, Maximum Entropy Markov Models (or MEMMs)
 - Conditional Random Fields (or CRFs).

Generative vs Discriminative Models

- Generative Models specify a *joint* distribution over the labels and the data. e.g. HMMs

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$$

- Discriminative models compute the *conditional* distribution of the labels given the input. You want to **discriminate** between different labels.

$$p(\mathbf{y}|\mathbf{x})$$

Discriminative Model

- General form:

$$p(\mathbf{y}|\mathbf{x})$$

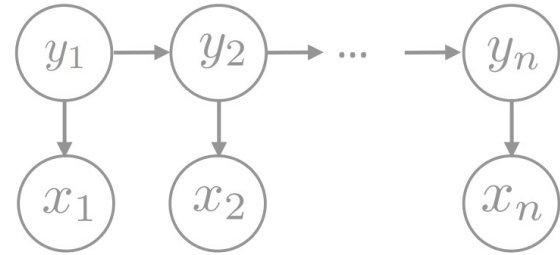
- Zero-th order Markov assumption

$$\prod_{i=1}^n p(y_i|\mathbf{x})$$

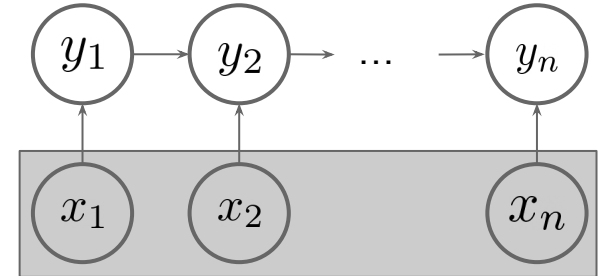
- First order Markov assumption

$$\prod_{i=1}^n p(y_i|y_{i-1}, \mathbf{x})$$

HMM



MEMM



Maximum Entropy Markov Models

t_1, t_2, \dots, t_C

$$\prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})$$

A text classification problem
(with more than 2 classes)

multiclass logistic regression

Define a score function $s(y_i, y_{i-1}, \mathbf{x})$

$$p(y_i = t_j | y_{i-1}, \mathbf{x}) \propto e^{s(y_i=t_j, y_{i-1}, \mathbf{x})}$$

$$p(y_i = t_j | y_{i-1}, \mathbf{x}) = \frac{e^{s(y_i=t_j, y_{i-1}, \mathbf{x})}}{\sum_{k=1}^C e^{s(y_i=t_k, y_{i-1}, \mathbf{x})}}$$

Normalization

Scoring Function

$$s(y_i, y_{i-1}, \mathbf{X}) = \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{X})$$

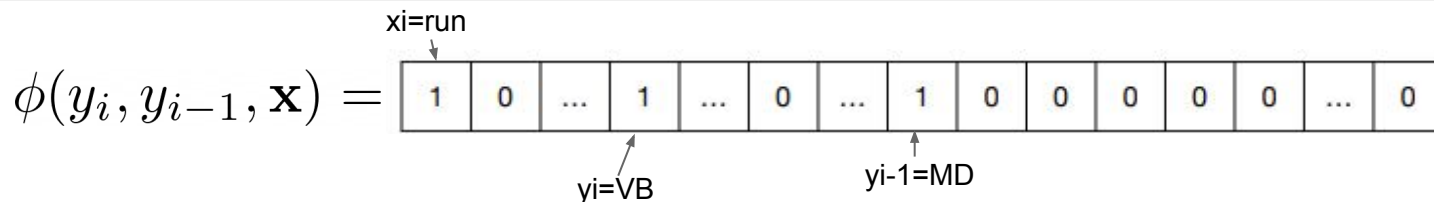
weights
(vector)

Feature function
(vector)

Three Questions

1. How to define features?
2. How to learn the weights for features?
3. How to perform inference (decoding)?

How to define features



- What is the current word, x_i ?
 - Number of features: size of vocabulary

- What is the previous label y_{i-1} ?
 - Number of features: total number of tags

- What is the previous word x_{i-1} ... ?
 - Number of features: size of vocabulary

Example: I will **run**.

More interesting features

- Is the current word capitalized?
- Does the current (or previous) word end in -ly, -ed, ...
- Does the current word contain digits, or a period?

I will absolutely **friend** you on **Facebook**.

Features can also be learned

Using neural networks.

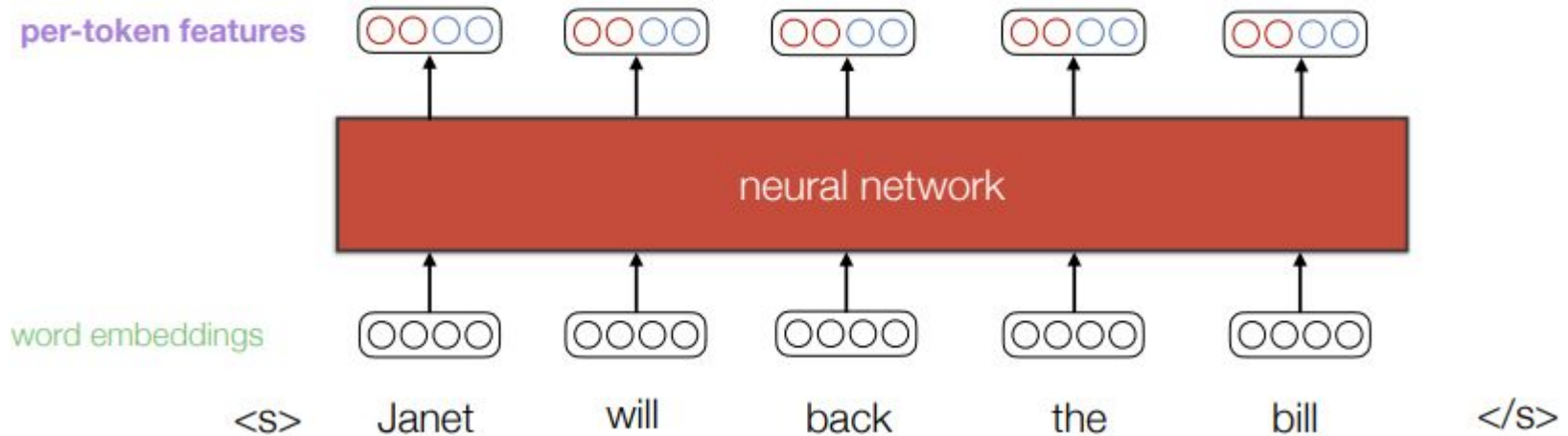
Basic Idea: encode the sequence of words into a sequence of vectors



Features can also be learned

Using neural networks.

Basic Idea: encode the sequence of words into a sequence of vectors



Features can also be learned

How to encode: Recurrent Neural Networks

$$h_i = F_{\theta}(h_{i-1}, x_i) \quad h_0 = \mathbf{0}$$

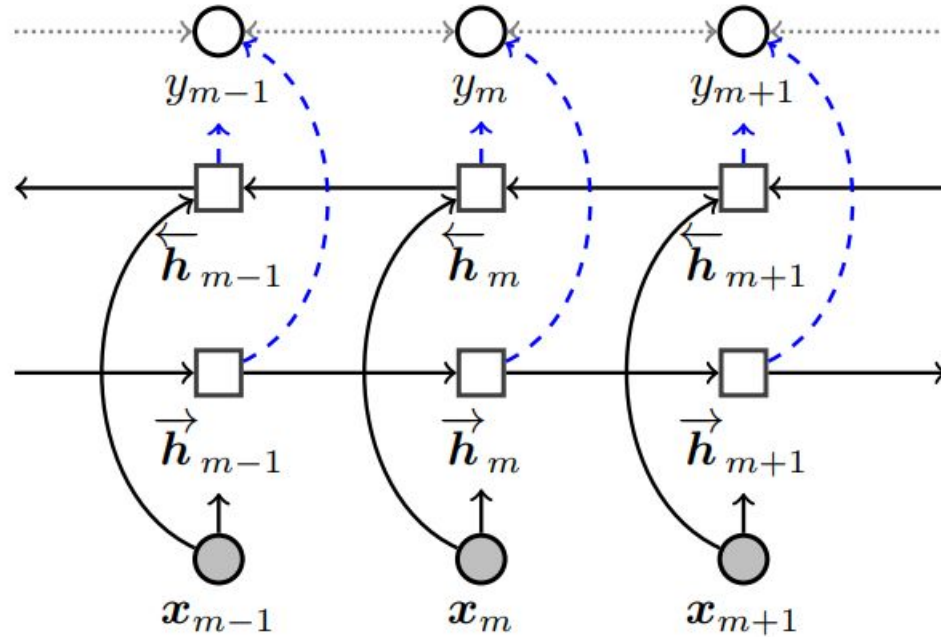
Token Feature at i
Token Feature at i-1
Input at i

$$\phi(y_i, y_{i-1}, \mathbf{x}) = G(h_i, y_i, y_{i-1})$$

What is F:

- LSTM
- Transformers
- ...

Bi-RNN-CRF



How to estimate the weights?

$$p(y_i | y_{i-1}, \mathbf{x}) = \frac{e^{s(y_i, y_{i-1}, \mathbf{x})}}{\sum_{i=1}^C e^{s(y_i, y_{i-1}, \mathbf{x})}} \quad s(y_i, y_{i-1}, \mathbf{x}) = \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})$$

- Supervised Classification (text and labels are provided as training data)

Input $\mathbf{x} = (x_1, \dots, x_n)$ Output $\mathbf{y} = (y_1, \dots, y_n)$

- Minimize cross entropy (aka Negative Log Likelihood) to find weights w and (also neural network parameters).

$$\sum -\log p(y_i | y_{i-1}, \mathbf{x})$$

How to decode? Viterbi again!

$$\arg \max_{\mathbf{y}} \prod_{i=1}^n p(y_i | y_{i-1}) p(y_i | x_i) \longrightarrow \arg \max_{\mathbf{y}} \prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})$$

HMM MEMM

$$\text{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \text{score}_{i-1}(y_{i-1}) \longrightarrow \text{score}_i(s) = \max_{y_{i-1}} p(s | y_{i-1}, \mathbf{x}) \text{score}_{i-1}(y_{i-1})$$

Viterbi decoding with HMMs Viterbi decoding with MEMMs

How to decode?

$$\arg \max_{\mathbf{y}} \prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})$$

$$\max_{y_1, \dots, y_n} p(y_1 | y_0, \mathbf{x}) p(y_2 | y_1, \mathbf{x}) \dots p(y_n | y_{n-1}, \mathbf{x})$$

Say I knew the value of $y_{n-1} = u$

$$\text{score}_n(y_{n-1} = s) = \max_{(y_1, \dots, y_{n-2})} p(y_1 | y_0, \mathbf{x}) \dots p(y_{n-1} = s | y_{n-2}, \mathbf{x})$$

$$\text{score}_n(y_n = s') = \max_{y_{n-1}} p(y_n = s' | y_{n-1}, \mathbf{x}) \text{score}_{n-1}(y_{n-1})$$

This recurrence relation can be solved by dynamic programming. AKA Viterbi algorithm

MEMMs - Summary

$$\prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})$$

$$p(y_i | y_{i-1}, \mathbf{x}) = \frac{e^{s(y_i, y_{i-1}, \mathbf{x})}}{\sum_{i=1}^C e^{s(y_i, y_{i-1}, \mathbf{x})}} \quad s(y_i, y_{i-1}, \mathbf{x}) = \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})$$

- Training – Cross Entropy Loss and Gradient Descent

- Decoding: Viterbi Algorithm $\arg \max_{\mathbf{y}} \prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})$

Local Normalization

- MEMMs are **locally normalized**.

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n p(y_i|y_{i-1}, \mathbf{x})$$

Conditional distribution
sums to one for each step i .

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n \frac{e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}}{\sum_{i=1}^C e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}}$$

Local Normalization to Global Normalization

Conditional Random Fields

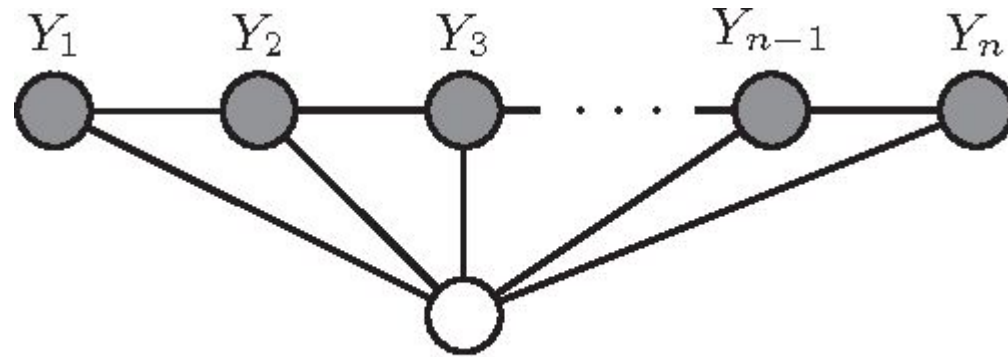
- If we do **global normalization**, we get “conditional random fields” or CRFs.

$$\prod_{i=1}^n \frac{e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}}{\sum_{i=1}^C e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}} \longrightarrow p(\mathbf{y} | \mathbf{x}) = \frac{e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}}{\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}}$$

$$\text{Global Feature vector} \quad \Phi(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x}) \quad \text{Old feature vector}$$

CRF - Undirected Graphical Model

$$\Phi(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x})$$



$$\mathbf{X} = X_1, \dots, X_{n-1}, X_n$$

CRFs

- How to find the weights – training?
 - Same as MEMM, minimize cross entropy
 - We have the same set of weights we had with MEMM. How they are learned is different (and better).

- How to find tags at test time – decoding?
 - Viterbi

Decoding with CRFs - Viterbi Algorithm

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \frac{e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}}{\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \Phi(\mathbf{y}', \mathbf{x})}}$$

Independent of \mathbf{y}

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}$$

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})$$

Decoding with CRFs - Viterbi Algorithm

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})$$

$$\Phi(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x})$$

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \mathbf{w} \cdot \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x})$$

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \sum_{i=1}^n \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})$$

Decoding with CRFs - Viterbi Algorithm

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \sum_{i=1}^n \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})$$

This can be also solved with Viterbi!

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

Viterbi decoding with HMMs



$$\text{score}_i(s) = \max_{y_{i-1}} \mathbf{w} \cdot \phi(s, y_{i-1}, \mathbf{x}) + \text{score}_{i-1}(y_{i-1})$$

Viterbi decoding with CRFs

Training CRF weights

Input $\mathbf{x} = (x_1, \dots, x_n)$ Output $\mathbf{y} = (y_1, \dots, y_n)$

- Supervised Classification (text and labels are provided as training data)

$$\mathcal{L}(\mathbf{w}) = -\log p(\mathbf{y}|\mathbf{x})$$

$$\mathcal{L}(\mathbf{w}) = -\log \frac{e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}}{\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \Phi(\mathbf{y}', \mathbf{x})}}$$

Really expensive!

The normalization complexity is huge — every possible sequence of labels of length n

Can solve fast by dynamic programming!

$$\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \Phi(\mathbf{y}', \mathbf{x})}$$

$$\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x})}$$

$$\sum_{\mathbf{y}'} e^{\sum_{i=1}^n \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}$$

Forward Algorithm

$$\sum_{\mathbf{y}'} e^{\sum_{i=1}^n \mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}$$

$$\sum_{\mathbf{y}'} \prod_{i=1}^n e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}$$

$$\boxed{\arg \max_{\mathbf{y}} \prod_{i=1}^n p(y_i | y_{i-1}, \mathbf{x})}$$

Both terms compute a product of some values dependent of y_i , y_{i-1} , and \mathbf{x}

The max is replaced with sum

Dynamic Programming: Forward Algorithm

$$\sum_{\mathbf{y}'} \prod_{i=1}^n e^{\mathbf{w} \cdot \phi(y_i, y_{i-1}, \mathbf{x})}$$

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

Viterbi decoding with HMMs



$$\text{score}_i(s) = \sum_{y_{i-1}} e^{\mathbf{w} \cdot \phi(s, y_{i-1}, \mathbf{x})} \text{score}_{i-1}(y_{i-1})$$

Forward algorithm in CRFs

To summarize:

CRFs:

$$p(\mathbf{y}|\mathbf{x}) = \frac{e^{\mathbf{w} \cdot \Phi(\mathbf{y}, \mathbf{x})}}{\sum_{\mathbf{y}'} e^{\mathbf{w} \cdot \Phi(\mathbf{y}', \mathbf{x})}}$$

$$\Phi(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^n \phi(y_i, y_{i-1}, \mathbf{x})$$

Features:

- Hand engineered or based on neural networks.

Training:

- Cross Entropy Loss (and Gradient Descent) + Forward Algorithm

Decoding

- Viterbi Algorithm

Readings

- Log Linear Models, MEMMs and CRFS (Michael Collins): [crf.pdf \(columbia.edu\)](#)
- BiLSTM-CRFs for sequence labeling: [\[1508.01991\] Bidirectional LSTM-CRF Models for Sequence Tagging \(arxiv.org\)](#)
- Natural Language Processing, Jacob Eisenstein (7.5.3): [eisenstein-nov18.pdf \(ucsd.edu\)](#)