Learning Structured Predictors

Xavier Carreras

DMETRICS

https://dmetrics.com

Supervised (Structured) Prediction

- Learning to predict: given training data

$$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$$

  learn a predictor $\mathbf{x} \to \mathbf{y}$ that *works well* on unseen inputs $\mathbf{x}$

- Non-Structured Prediction: outputs $\mathbf{y}$ are atomic
  - Binary prediction: $\mathbf{y} \in \{-1, +1\}$
  - Multiclass prediction: $\mathbf{y} \in \{1, 2, \ldots, L\}$

- Structured Prediction: outputs $\mathbf{y}$ are structured
  - Sequence prediction: $\mathbf{y}$ are sequences
  - Parsing: $\mathbf{y}$ are trees
  - ...

Named Entity Recognition

$\mathbf{y}$	PER	-	QNT	-	-	ORG	ORG	-	TIME
$\mathbf{x}$	Jim	bought	300	shares	of	Acme	Corp.	in	2006

Named Entity Recognition

$\mathbf{y}$	PER	-	QNT	-	-	ORG	ORG	-	TIME
$\mathbf{x}$	Jim	bought	300	shares	of	Acme	Corp.	in	2006

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

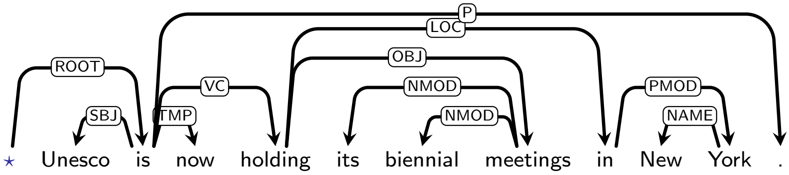
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Paris	Hilton	went	to	London

$\mathbf{y}$	PER	-	-	LOC
$\mathbf{x}$	Jackie	went	to	Lisdon

Part-of-speech Tagging

$\mathbf{y}$	NNP	NNP	VBZ	NNP	.
$\mathbf{x}$	Ms.	Haag	plays	Elianti	.

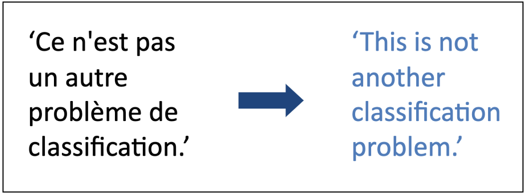
Syntactic Dependency Parsing



$\mathbf{x}$ are sentences

$\mathbf{y}$ are syntactic dependency trees

Machine Translation



(illustration by Ben Taskar)

$\mathbf{x}$ are sentences in some source language (e.g. French)
$\mathbf{y}$ are sentence translations in a target language (e.g. English)

Object Detection

(Kumar and Hebert, 2003)

$\mathbf{x}$ are images

$\mathbf{y}$ are grids labeled with object types

Object Detection

(Kumar and Hebert, 2003)

$\mathbf{x}$ are images

$\mathbf{y}$ are grids labeled with object types

Today's Goals

- Introduce basic concepts for structured prediction
  - We will focus on sequence prediction
- What can we can borrow from standard classification?
  - Learning paradigms and algorithms, in essence, work here too
  - However, computations behind algorithms are prohibitive
- What can we borrow from HMM and other structured formalisms?
  - Representations of structured data into feature spaces
  - Inference/search algorithms for tractable computations
  - E.g., algorithms for HMMs (Viterbi, forward-backward) will play a major role in today's methods

Today's Goals

- Introduce basic concepts for structured prediction
  - We will focus on sequence prediction

- What can we can borrow from standard classification?
  - Learning paradigms and algorithms, in essence, work here too
  - However, computations behind algorithms are prohibitive

- What can we borrow from HMM and other structured formalisms?
  - Representations of structured data into feature spaces
  - Inference/search algorithms for tractable computations
  - E.g., algorithms for HMMs (Viterbi, forward-backward) will play a major role in today's methods

Sequence Prediction

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

Sequence Prediction

- $\mathbf{x} = x_1 x_2 \ldots x_n$ are input sequences, $x_i \in \mathcal{X}$
- $\mathbf{y} = y_1 y_2 \ldots y_n$ are output sequences, $y_i \in \{1, \ldots, L\}$

- **Goal:** given training data

$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$

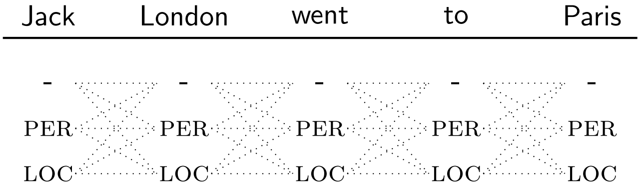
  learn a predictor $\mathbf{x} \to \mathbf{y}$ that works well on unseen inputs $\mathbf{x}$

- What is the form of our prediction model?

Exponentially-many Solutions

- Let $\mathcal{Y} = \{\text{-}, \textsc{per}, \textsc{loc}\}$

- The solution space (all output sequences):

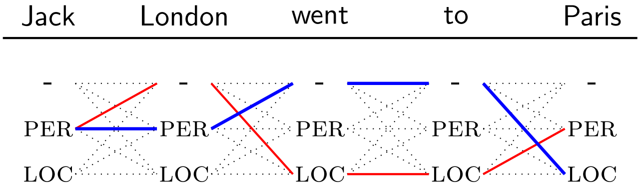


  - Each path is a possible solution

- For an input sequence of size n, there are $|\mathcal{Y}|^n$ possible outputs

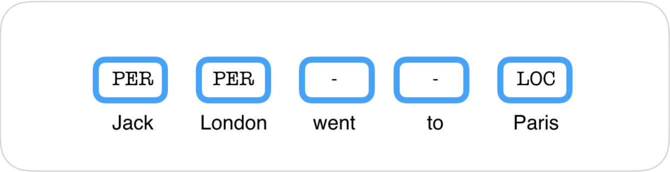
Exponentially-many Solutions

- Let $\mathcal{Y} = \{\text{-}, \text{PER}, \text{LOC}\}$
- The solution space (all output sequences):



  - Each path is a possible solution
- For an input sequence of size n, there are $|\mathcal{Y}|^n$ possible outputs

Approach 1: Label Classifiers



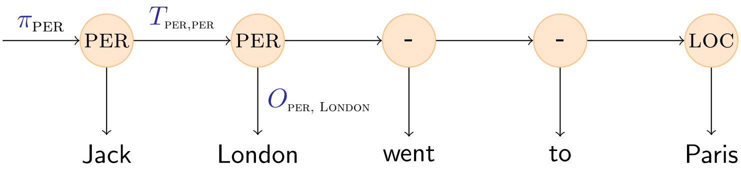
- Multiclass prediction over individual labels at each position

$$\hat{y}_i = \underset{l \in \{\text{LOC},\ \text{PER},\ \text{-}\}}{\operatorname{argmax}} \quad \text{score}(\mathbf{x}, i, l)$$

- For linear models, $\text{score}(\mathbf{x}, i, l) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, l)$
  - $\mathbf{f}(\mathbf{x}, i, l) \in \mathbb{R}^d$ represents an assignment of label l for x_i
  - $\mathbf{w} \in \mathbb{R}^d$ is a vector of parameters (learned), has a weight for each feature in $\mathbf{f}$
- Can capture interactions between full input sequence $\mathbf{x}$ and one output label l

  e.g.: current word, surrounding words, capitalization, prefix-suffix, gazetteer, ...
- Can not capture interactions between output labels!

Approach 2: HMM for Sequence Prediction

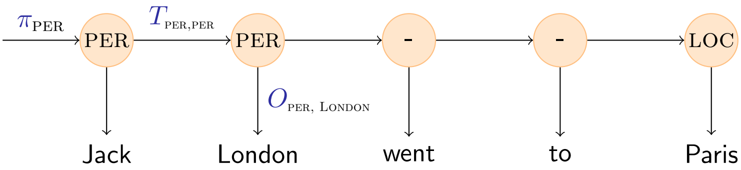


- Define an HMM where each label is a state
- Model parameters:
  - π_l : probability of starting with label l
  - $T_{l,l'}$: probability of transitioning from label l to l'
  - $O_{l,x}$: probability of generating symbol x given label l
- Probability distribution:

$$p(\mathbf{x}, \mathbf{y}) = \pi_{y_1} O_{y_1,x_1} \prod_{i>1} T_{y_{i-1},y_i} O_{y_i,x_i}$$

- Learning: relative counts + smoothing (or EM in case of missing labeled data)
- Prediction: Viterbi algorithm

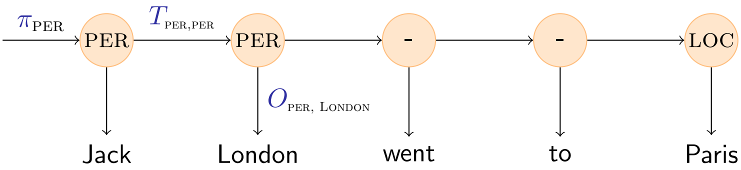
Approach 2: Representation in HMM



- Label interactions are captured in the transition parameters
- But interactions between labels and input symbols are very limited!
  - Only $O_{y_i, x_i} = p(x_i \mid y_i)$
  - Not clear how to exploit patterns such as:
    - Sub-symbol: capitalization, digits, prefixes, suffixes, ...
    - Context: previous word, next word, ...
    - Combinations of these with label transitions
- Why? HMM independence assumptions:

  given label y_i, token x_i is independent of anything else

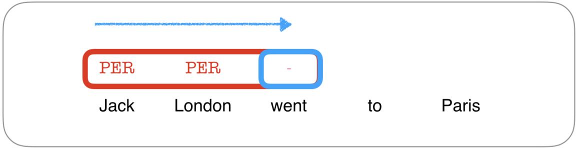
Approach 2: Representation in HMM



- Label interactions are captured in the transition parameters
- But interactions between labels and input symbols are very limited!
  - Only $O_{y_i,x_i} = p(x_i \mid y_i)$
  - Not clear how to exploit patterns such as:
    - Sub-symbol: capitalization, digits, prefixes, suffixes, ...
    - Context: previous word, next word, ...
    - Combinations of these with label transitions
- Why? HMM independence assumptions:

  given label y_i, token x_i is independent of anything else

Approach 3: Transition-based Sequence Prediction

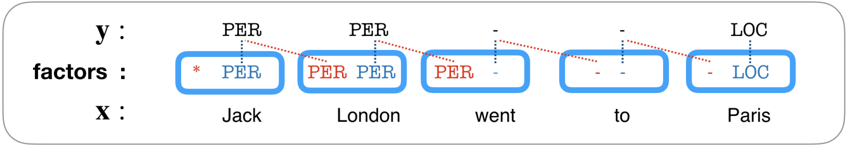


- Predict one label at a time, left-to-right, using previous predictions:

$$\hat{y}_i = \underset{l \in \{\text{LOC}, \text{PER}, \text{-}\}}{\operatorname{argmax}} \operatorname{score}(\mathbf{x}, i, l, \hat{\mathbf{y}}_{1:i-1})$$

- Captures interactions between full input $\mathbf{x}$ and prefixes of the output sequence
- Prediction of $\hat{\mathbf{y}}$ is approximate (greedy, beam search)
  - Why left-to-right and not right-to-left?

Approach 4: Factored Sequence Prediction



- At each position, multiclass prediction over label bigrams (pairs of adjacent labels):

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\,\in\,\mathcal{Y}^n}{\operatorname{argmax}} \operatorname{score}(\mathbf{x}, \mathbf{y}) = \underset{\mathbf{y}\,\in\,\mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \operatorname{score}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Output sequence factored into label bigrams
- Captures interactions between full input $\mathbf{x}$ and factors of output sequence
- Prediction is tractable for many types of factorizations (this lecture)

Approach 5: Re-Ranking

PER	PER	-	-	LOC
PER	LOC	-	-	LOC
LOC	LOC	-	-	LOC
PER	PER	-	-	PER
PER	PER	PER	-	LOC
...	...	...	...	...
Jack	London	went	to	Paris

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{A}(\mathcal{Y}^n)} \operatorname{score}(\mathbf{x}, \mathbf{y})$$

- Scoring of full inputs and outputs: very expressive!
- Relies on an active set $\mathcal{A}(\mathcal{Y}^n)$ of full outputs, enumerated exhaustively
- A base model is used to select active set
  - The base model follows one of the previous approaches

Sequence Prediction: Summary of Approaches

	input-output representation	exact prediction?
label classifiers	only individual labels	yes
HMM	label factors but limited input	yes
transition-based	full history of decisons	no (greedy, beam search)
factored	label factors	yes
re-ranking	full	limited to active set

take home message 1: the expressivity-tractability trade-off exists

take home message 2: always pick the simplest approach that suits the task at hand

Sequence Prediction: Summary of Approaches

	input-output representation	exact prediction?
label classifiers	only individual labels	yes
HMM	label factors but limited input	yes
transition-based	full history of decisons	no (greedy, beam search)
factored	label factors	yes
re-ranking	full	limited to active set

take home message 1: the expressivity-tractability trade-off exists

take home message 2: always pick the simplest approach that suits the task at hand

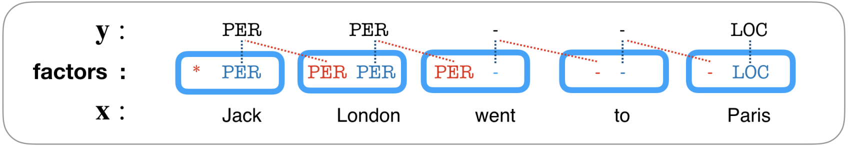
Sequence Prediction: Summary of Approaches

	input-output representation	exact prediction?
label classifiers	only individual labels	yes
HMM	label factors but limited input	yes
transition-based	full history of decisons	no (greedy, beam search)
factored	label factors	yes
re-ranking	full	limited to active set

take home message 1: the expressivity-tractability trade-off exists

take home message 2: always pick the simplest approach that suits the task at hand

Factored Sequence Predictors



$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \operatorname{score}(\mathbf{x}, i, y_{i-1}, y_i)$$

Next questions:

- There are exponentially-many sequences $\mathbf{y}$ for a given $\mathbf{x}$, how do we solve the argmax problem?
- What is the form of $\operatorname{score}(\mathbf{x}, i, a, b)$?
  We will use linear scoring functions: $\operatorname{score}(\mathbf{x}, i, a, b) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)$
- How do we learn $\mathbf{w}$?

Predicting with Factored Sequence Models

- Assume we have a score function $\mathrm{score}(\mathbf{x}, i, a, b)$
- Given $\mathbf{x}_{1:n}$ find:

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \sum_{i=1}^{n} \mathrm{score}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Use the Viterbi algorithm, takes $O(n|\mathcal{Y}|^2)$
- Notational change: since $\mathbf{x}_{1:n}$ is fixed we will use

$$s(i, a, b) = \mathrm{score}(\mathbf{x}, i, a, b)$$

Viterbi for Factored Sequence Models

- Given scores $s(i, a, b)$ for each position i and output bigram a, b, find:

$$\underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} s(i, y_{i-1}, y_i)$$

- Intuition: consider this example $\mathbf{x}$ and two alternative solutions $\mathbf{y}$ and $\mathbf{y}'$:

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	LOC	-	-	LOC
$\mathbf{y}'$	PER	PER	-	-	LOC

- What is the score of y' relative to the score of y?

$$\begin{aligned} s(\mathbf{x}, \mathbf{y}') &= s(\mathbf{x}, \mathbf{y}) \quad + \qquad\qquad - \\ &\qquad\qquad\quad + \qquad\qquad - \end{aligned}$$

Viterbi for Factored Sequence Models

- Given scores $s(i, a, b)$ for each position i and output bigram a, b, find:

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \sum_{i=1}^{n} s(i, y_{i-1}, y_i)$$

- Intuition: consider this example $\mathbf{x}$ and two alternative solutions $\mathbf{y}$ and $\mathbf{y}'$:

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	LOC	-	-	LOC
$\mathbf{y}'$	PER	PER	-	-	LOC

- What is the score of y' relative to the score of y?

$$\begin{aligned} s(\mathbf{x}, \mathbf{y}') \quad = s(\mathbf{x}, \mathbf{y}) \quad & + s(2, \text{PER}, \text{ PER}) - s(2, \text{PER}, \text{ LOC}) \\ & + s(3, \text{LOC}, \text{ -}) - s(3, \text{PER}, \text{ -}) \end{aligned}$$

output sequences that share bigrams also share scores

Intuition for Viterbi

- Assume that, for each label $l \in \mathcal{Y}$, we have the best sub-sequence from positions 1 to i ending with label l:

$1 \quad \cdots \quad i \quad\quad i+1$

best subsequence with $y_i = \text{PER}$

best subsequence with $y_i = \text{LOC}$

best subsequence with $y_i = -$

- What is the best sequence up to position $i+1$ with y_{i+1} $=\text{LOC}$?

Intuition for Viterbi

- Assume that, for each label $l \in \mathcal{Y}$, we have the best sub-sequence from positions 1 to i ending with label l:

$1 \quad \cdots \quad i \quad i+1$

best subsequence with $y_i = \text{PER}$

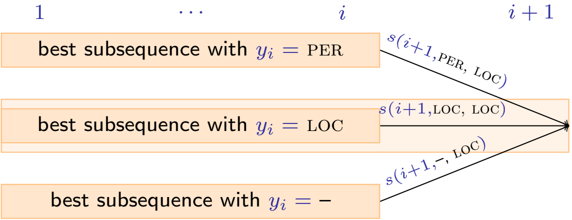
best subsequence with $y_i = \text{LOC}$

best subsequence with $y_i = \text{–}$

- What is the best sequence up to position $i+1$ with $y_{i+1} = \text{LOC}$?

Intuition for Viterbi

- Assume that, for each label $l \in \mathcal{Y}$, we have the best sub-sequence from positions 1 to i ending with label l:



- What is the best sequence up to position $i+1$ with $y_{i+1} = $LOC?

Viterbi for Factored Sequence Models

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \sum_{i=1}^{n} s(i, y_{i-1}, y_i)$$

- **Definition:** score of optimal sequence for $\mathbf{x}_{1:i}$ ending with $a \in \mathcal{Y}$

$$\delta(i, a) = \max_{\mathbf{y} \in \mathcal{Y}^i : y_i = a} \sum_{j=1}^{i} s(j, y_{j-1}, y_j)$$

- Use the following recursions, for all $a \in \mathcal{Y}$:

$$\begin{aligned} \delta(1, a) &= s(1, y_0 = \text{NULL}, a) \\ \delta(i, a) &= \max_{b \in \mathcal{Y}} \delta(i-1, b) + s(i, b, a) \end{aligned}$$

- The optimal score for $\mathbf{x}$ is $\max_{a \in \mathcal{Y}} \delta(n, a)$
- The optimal sequence $\hat{\mathbf{y}}$ can be recovered through *back-pointers*
- Homework: rewrite the Viterbi equations such that the algorithm proceeds right-to-left. Observe that the factored model remains the same (i.e. it is not a directional model)

Viterbi for Factored Sequence Models

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \sum_{i=1}^{n} s(i, y_{i-1}, y_i)$$

- **Definition:** score of optimal sequence for $\mathbf{x}_{1:i}$ ending with $a \in \mathcal{Y}$

$$\delta(i, a) = \max_{\mathbf{y} \in \mathcal{Y}^i : y_i = a} \sum_{j=1}^{i} s(j, y_{j-1}, y_j)$$

- Use the following recursions, for all $a \in \mathcal{Y}$:

$$\begin{aligned} \delta(1, a) &= s(1, y_0 = \text{NULL}, a) \\ \delta(i, a) &= \max_{b \in \mathcal{Y}} \delta(i-1, b) + s(i, b, a) \end{aligned}$$

- The optimal score for $\mathbf{x}$ is $\max_{a \in \mathcal{Y}} \delta(n, a)$
- The optimal sequence $\hat{\mathbf{y}}$ can be recovered through *back-pointers*
- Homework: rewrite the Viterbi equations such that the algorithm proceeds right-to-left. Observe that the factored model remains the same (i.e. it is not a directional model)

Linear Factored Models and Representations

$$\underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \operatorname{score}(\mathbf{x}, i, y_{i-1}, y_i)$$

- In linear factored models:

$$\operatorname{score}(\mathbf{x}, i, a, b) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)$$

- $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector, to be learned
- $\mathbf{f}(\mathbf{x}, i, a, b) \in \mathbb{R}^d$ is a feature vector
- How to construct $\mathbf{f}(\mathbf{x}, i, a, b)$?
  - New trend: representation learning
  - Old school: manually with feature templates

Linear Factored Models and Representations

$$\underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \operatorname{score}(\mathbf{x}, i, y_{i-1}, y_i)$$

- In linear factored models:

$$\operatorname{score}(\mathbf{x}, i, a, b) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)$$

- $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector, to be learned
- $\mathbf{f}(\mathbf{x}, i, a, b) \in \mathbb{R}^d$ is a feature vector
- How to construct $\mathbf{f}(\mathbf{x}, i, a, b)$?
  - New trend: representation learning
  - Old school: manually with feature templates

Indicator Features for Label Unigrams

- $\mathbf{f}(\mathbf{x}, i, l)$ is a vector of d features representing label l for x_i

$$[\ \mathbf{f}_1(\mathbf{x}, i, l), \ldots, \mathbf{f}_j(\mathbf{x}, i, l), \ldots, \mathbf{f}_d(\mathbf{x}, i, l)\ ]$$

- What's in a feature $\mathbf{f}_j(\mathbf{x}, i, l)$?
  - Anything we can compute using $\mathbf{x}$ and i and l
  - Anything that indicates whether l is (not) a good label for x_i
  - Indicator features: binary-valued features looking at:
    - a simple pattern of $\mathbf{x}$ and target position i
    - and the candidate label l for position i

$$\mathbf{f}_j(\mathbf{x}, i, l) = \begin{cases} 1 & \text{if } x_i = \text{London and } l = \textsc{loc} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_k(\mathbf{x}, i, l) = \begin{cases} 1 & \text{if } x_{i+1} = \text{went and } l = \textsc{loc} \\ 0 & \text{otherwise} \end{cases}$$

Feature Templates

- Feature templates generate many indicator features mechanically
- A feature template is identified by a type, and a number of values
  - Example: template WORD indicates the current word

$$\mathbf{f}_{\langle \textsc{word}, a, w \rangle}(\mathbf{x}, i, l) = \begin{cases} 1 & \text{if } x_i = w \text{ and } l = a \\ 0 & \text{otherwise} \end{cases}$$

  - A feature of this type is identified by the tuple $\langle \textsc{word}, a, w \rangle$
  - Generates a feature for every label $a \in \mathcal{Y}$ and every word w

e.g.:	$a = \textsc{loc}$	$w =$ London,	$a =$ -	$w =$ London
	$a = \textsc{loc}$	$w =$ Paris	$a = \textsc{per}$	$w =$ Paris
	$a = \textsc{per}$	$w =$ John	$a =$ -	$w =$ the

Feature Templates

- Feature templates generate many indicator features mechanically
- A feature template is identified by a type, and a number of values
  - Example: template $\textsc{word}$ indicates the current word

$$\mathbf{f}_{\langle\textsc{word},a,w\rangle}(\mathbf{x},i,l) = \begin{cases} 1 & \text{if } x_i = w \text{ and } l = a \\ 0 & \text{otherwise} \end{cases}$$

  - A feature of this type is identified by the tuple $\langle\textsc{word}, a, w\rangle$
  - Generates a feature for every label $a \in \mathcal{Y}$ and every word w

    e.g.: $a = \textsc{loc}$ $w =$ London, $a =$ - $w =$ London
    $a = \textsc{loc}$ $w =$ Paris $a = \textsc{per}$ $w =$ Paris
    $a = \textsc{per}$ $w =$ John $a =$ - $w =$ the
- In feature-based models:
  - Define feature templates manually
  - Instantiate the templates on every set of values in the training data
    $\rightarrow$ generates a very high-dimensional feature space
  - Define parameter vector $\mathbf{w}$ indexed by such feature tuples
  - Let the learning algorithm choose the relevant features

More Features for NE Recognition

$$\begin{array}{ccccc} & \text{PER} & & & \\ \text{Jack} & \text{London} & \text{went} & \text{to} & \text{Paris} \end{array}$$

In practice, construct $\mathbf{f}(\mathbf{x}, i, l)$ by ...

- Define a number of simple patterns of $\mathbf{x}$ and i
  - current word x_i
  - is x_i capitalized?
  - x_i has digits?
  - prefixes/suffixes of size 1, 2, 3, ...
  - is x_i a known location?
  - is x_i a known person?
  - next word
  - previous word
  - current and next words together
  - other combinations
- Define feature templates by combining patterns with labels l
- Generate actual features by instantiating templates on training data

Bigram Feature Templates

	1	2	3	4	5
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

- Example: A template for word + bigram:

$$\mathbf{f}_{\langle \mathrm{WB}, a, b, w \rangle}(\mathbf{x}, i, y_{i-1}, y_i) = \begin{cases} 1 & \text{if } x_i = w \text{ and} \\ & \quad y_{i-1} = a \text{ and } y_i = b \\ 0 & \text{otherwise} \end{cases}$$

e.g., $\mathbf{f}_{\langle \mathrm{WB}, \mathrm{PER}, \mathrm{PER}, \text{London} \rangle}(\mathbf{x}, 2, \mathrm{PER}, \mathrm{PER}) = 1$

$\mathbf{f}_{\langle \mathrm{WB}, \mathrm{PER}, \mathrm{PER}, \text{London} \rangle}(\mathbf{x}, 3, \mathrm{PER}, \text{-}) = 0$

$\mathbf{f}_{\langle \mathrm{WB}, \mathrm{PER}, \text{-}, \text{went} \rangle}(\mathbf{x}, 3, \mathrm{PER}, \text{-}) = 1$

- Bigram feature templates are strictly more expressive than unigram templates

More Templates for NER

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_{\langle \text{W},\text{PER},\text{PER},\text{London}\rangle}(\dots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_{\langle \text{W},\text{PER},\text{LOC},\text{London}\rangle}(\dots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{PREP},\text{LOC},\text{to}\rangle}(\dots) = 1$ iff $x_{i-1} =$"to" and $x_i \sim$/[A-Z]/ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{CITY},\text{LOC}\rangle}(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_{\langle \text{FNAME},\text{PER}\rangle}(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Templates for NER

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_{\langle \text{W,PER,PER,London} \rangle}(\ldots) = 1$ iff $x_i =$ "London" and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_{\langle \text{W,PER,LOC,London} \rangle}(\ldots) = 1$ iff $x_i =$ "London" and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{PREP,LOC,to} \rangle}(\ldots) = 1$ iff $x_{i-1} =$ "to" and $x_i \sim$/[A-Z]/ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{CITY,LOC} \rangle}(\ldots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_{\langle \text{FNAME,PER} \rangle}(\ldots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Templates for NER

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_{\langle \text{W,PER,PER,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_{\langle \text{W,PER,LOC,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{PREP,LOC,to} \rangle}(\ldots) = 1$ iff $x_{i-1} =$"to" and $x_i \sim$/[A-Z]/ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{CITY,LOC} \rangle}(\ldots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_{\langle \text{FNAME,PER} \rangle}(\ldots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Templates for NER

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_{\langle \text{W,PER,PER,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_{\langle \text{W,PER,LOC,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{PREP,LOC,to} \rangle}(\ldots) = 1$ iff $x_{i-1} =$"to" and $x_i \sim$/[A-Z]/ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{CITY,LOC} \rangle}(\ldots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_{\langle \text{FNAME,PER} \rangle}(\ldots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Templates for NER

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_{\langle \text{W,PER,PER,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_{\langle \text{W,PER,LOC,London} \rangle}(\ldots) = 1$ iff $x_i =$"London" and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{PREP,LOC,to} \rangle}(\ldots) = 1$ iff $x_{i-1} =$"to" and $x_i \sim$/[A-Z]/ and $y_i = \text{LOC}$

$\mathbf{f}_{\langle \text{CITY,LOC} \rangle}(\ldots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_{\langle \text{FNAME,PER} \rangle}(\ldots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

Representations Factored at Bigrams

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- $\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
  - A d-dimensional feature vector of a label bigram at i
  - Each dimension is typically a boolean indicator (0 or 1)
- $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
  - A d-dimensional feature vector of the entire $\mathbf{y}$
  - Aggregated representation by summing bigram feature vectors
  - Each dimension is now a count of a feature pattern

Linear Factored Sequence Prediction

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$$

where

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

- Note the linearity of the expression:

$$\begin{aligned}
\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}) &= \mathbf{w}\cdot\sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \\
&= \sum_{i=1}^{n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \\
&= \sum_{i=1}^{n} \mathrm{score}(\mathbf{x}, i, y_{i-1}, y_i)
\end{aligned}$$

Linear Factored Sequence Prediction

$$\underset{\mathbf{y}\in\mathcal{Y}^n}{\operatorname{argmax}}\, \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$$

- Factored representation, e.g. based on bigrams
- Flexible, arbitrary features of full $\mathbf{x}$ and the factors
- Efficient prediction using Viterbi
- Next, learning $\mathbf{w}$:
  - Probabilistic log-linear models:
    - Local learning, *a.k.a.* Maximum-Entropy Markov Models
    - Global learning, *a.k.a.* Conditional Random Fields
  - Margin-based methods:
    - Structured Perceptron
    - Structured SVM

The Learner's Game

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

The Learner's Game

Training Data

- PER Maria, - is, - young
- LOC Lisbon, - is, - big
- PER Jack, - went, - to, LOC Lisbon
- LOC Argentina, - is, - bigger
- PER Jack, PER London, - went, - to, LOC South, LOC Pacific
- ORG Argentina, - played, - against, ORG Chile

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$$\mathbf{w}_{\langle \textsc{Lower}, \text{-} \rangle} = +1$$

The Learner's Game

Training Data

- PER - -
  Maria is young
- LOC - -
  Lisbon is big
- PER - - LOC
  Jack went to Lisbon
- LOC - -
  Argentina is bigger
- PER PER - - LOC LOC
  Jack London went to South Pacific
- ORG - - ORG
  Argentina played against Chile

Weight Vector $\mathbf{w}$

$$\mathbf{w}_{\langle \text{LOWER},\text{-} \rangle} = +1$$
$$\mathbf{w}_{\langle \text{UPPER},\text{PER} \rangle} = +1$$

The Learner's Game

Training Data

- PER Maria, - is, - young
- LOC Lisbon, - is, - big
- PER Jack, - went, - to, LOC Lisbon
- LOC Argentina, - is, - bigger
- PER Jack, PER London, - went, - to, LOC South, LOC Pacific
- ORG Argentina, - played, - against, ORG Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \textsc{Lower}, \text{-} \rangle} = +1$

$\cancel{\mathbf{w}_{\langle \textsc{Upper}, \textsc{per} \rangle} = +1}$

$\mathbf{w}_{\langle \textsc{Upper}, \textsc{loc} \rangle} = +1$

The Learner's Game

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$$\mathbf{w}_{\langle \text{LOWER},\text{-}\rangle} = +1$$
$$\cancel{\mathbf{w}_{\langle \text{UPPER},\text{PER}\rangle} = +1}$$
$$\mathbf{w}_{\langle \text{UPPER},\text{LOC}\rangle} = +1$$
$$\mathbf{w}_{\langle \text{WORD},\text{PER},\text{Maria}\rangle} = +2$$

The Learner's Game

Training Data

- PER Maria, - is, - young
- LOC Lisbon, - is, - big
- PER Jack, - went, - to, LOC Lisbon
- LOC Argentina, - is, - bigger
- PER Jack, PER London, - went, - to, LOC South, LOC Pacific
- ORG Argentina, - played, - against, ORG Chile

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \textsc{Lower}, \text{-} \rangle} = +1$

$\cancel{\mathbf{w}_{\langle \textsc{Upper}, \textsc{per} \rangle} = +1}$

$\mathbf{w}_{\langle \textsc{Upper}, \textsc{loc} \rangle} = +1$

$\mathbf{w}_{\langle \textsc{Word}, \textsc{per}, \text{Maria} \rangle} = +2$

$\mathbf{w}_{\langle \textsc{Word}, \textsc{per}, \text{Jack} \rangle} = +2$

The Learner's Game

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \text{Lower},\text{-} \rangle} = +1$

$\cancel{\mathbf{w}_{\langle \text{Upper},\text{per} \rangle} = +1}$

$\mathbf{w}_{\langle \text{Upper},\text{loc} \rangle} = +1$

$\mathbf{w}_{\langle \text{Word},\text{per},\text{Maria} \rangle} = +2$

$\mathbf{w}_{\langle \text{Word},\text{per},\text{Jack} \rangle} = +2$

$\mathbf{w}_{\langle \text{NextW},\text{per},\text{went} \rangle} = +2$

The Learner's Game

Training Data

- PER - -
  Maria is young
- LOC - -
  Lisbon is big
- PER - - LOC
  Jack went to Lisbon
- LOC - -
  Argentina is bigger
- PER PER - - LOC LOC
  Jack London went to South Pacific
- ORG - - ORG
  Argentina played against Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \textsc{Lower},\text{-}\rangle} = +1$

$\cancel{\mathbf{w}_{\langle \textsc{Upper},\textsc{per}\rangle} = +1}$

$\mathbf{w}_{\langle \textsc{Upper},\textsc{loc}\rangle} = +1$

$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\text{Maria}\rangle} = +2$

$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\text{Jack}\rangle} = +2$

$\mathbf{w}_{\langle \textsc{NextW},\textsc{per},\text{went}\rangle} = +2$

$\mathbf{w}_{\langle \textsc{NextW},\textsc{org},\text{played}\rangle} = +2$

The Learner's Game

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$\mathbf{w}_{\langle \text{LOWER},\text{-} \rangle} = +1$

$\cancel{\mathbf{w}_{\langle \text{UPPER},\text{PER} \rangle} = +1}$

$\mathbf{w}_{\langle \text{UPPER},\text{LOC} \rangle} = +1$

$\mathbf{w}_{\langle \text{WORD},\text{PER},\text{Maria} \rangle} = +2$

$\mathbf{w}_{\langle \text{WORD},\text{PER},\text{Jack} \rangle} = +2$

$\mathbf{w}_{\langle \text{NEXTW},\text{PER},\text{went} \rangle} = +2$

$\mathbf{w}_{\langle \text{NEXTW},\text{ORG},\text{played} \rangle} = +2$

$\mathbf{w}_{\langle \text{PREVW},\text{ORG},\text{against} \rangle} = +2$

The Learner's Game

Training Data

PER	-	-
Maria	is	young

LOC	-	-
Lisbon	is	big

PER	-	-	LOC
Jack	went	to	Lisbon

LOC	-	-
Argentina	is	bigger

PER	PER	-	-	LOC	LOC
Jack	London	went	to	South	Pacific

ORG	-	-	ORG
Argentina	played	against	Chile

Weight Vector $\mathbf{w}$

$$\mathbf{w}_{\langle \textsc{Lower},\text{-}\rangle} = +1$$
$$\cancel{\mathbf{w}_{\langle \textsc{Upper},\textsc{per}\rangle} = +1}$$
$$\mathbf{w}_{\langle \textsc{Upper},\textsc{loc}\rangle} = +1$$
$$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\text{Maria}\rangle} = +2$$
$$\mathbf{w}_{\langle \textsc{Word},\textsc{per},\text{Jack}\rangle} = +2$$
$$\mathbf{w}_{\langle \textsc{NextW},\textsc{per},\text{went}\rangle} = +2$$
$$\mathbf{w}_{\langle \textsc{NextW},\textsc{org},\text{played}\rangle} = +2$$
$$\mathbf{w}_{\langle \textsc{PrevW},\textsc{org},\text{against}\rangle} = +2$$
$$\ldots$$
$$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{per},\textsc{per}\rangle} = +100$$
$$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{loc},\textsc{loc}\rangle} = +100$$
$$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{loc},\textsc{per}\rangle} = -100$$
$$\mathbf{w}_{\langle \textsc{UpperBigram},\textsc{per},\textsc{loc}\rangle} = -100$$
$$\mathbf{w}_{\langle \textsc{NextW},\textsc{loc},\text{played}\rangle} = -1000$$

Log-linear Models

for Sequence Prediction

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

Log-linear Models for Sequence Prediction

- Model the conditional distribution:

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z(\mathbf{x}; \mathbf{w})}$$

  where

  - $\mathbf{x} = x_1 x_2 \ldots x_n \in \mathcal{X}^*$
  - $\mathbf{y} = y_1 y_2 \ldots y_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1, \ldots, L\}$
  - $\mathbf{f}(\mathbf{x}, \mathbf{y})$ represents $\mathbf{x}$ and $\mathbf{y}$ with d features
  - $\mathbf{w} \in \mathbb{R}^d$ are the parameters of the model
  - $Z(\mathbf{x}; \mathbf{w})$ is a normalizer called the *partition function*

$$Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{z} \in \mathcal{Y}^*} \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})\}$$

- To predict the best sequence

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \Pr(\mathbf{y} | \mathbf{x})$$

Log-linear Models: Name

- Let's take the $\log$ of the conditional probability:

$$\begin{aligned}
\log \Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) &= \log \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z(\mathbf{x}; \mathbf{w})} \\
&= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) - \log \sum_{y} \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\} \\
&= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) - \log Z(\mathbf{x}; \mathbf{w})
\end{aligned}$$

- Partition function: $Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{z}} \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})\}$
- $\log Z(\mathbf{x}; \mathbf{w})$ is a constant for a fixed $\mathbf{x}$
- In the $\log$ space, computations are linear,
  i.e., we model log-probabilities using a linear predictor

Making Predictions with Log-Linear Models

- For tractability, assume $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes into bigrams:

$$\mathbf{f}(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Given $\mathbf{w}$, given $\mathbf{x}_{1:n}$, find:

$$\begin{aligned}
\underset{\mathbf{y}_{1:n}}{\operatorname{argmax}} \Pr(\mathbf{y}_{1:n} | \mathbf{x}_{1:n}; \mathbf{w}) &= \underset{\mathbf{y}}{\operatorname{amax}} \frac{\exp\left\{\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\right\}}{Z(\mathbf{x}; \mathbf{w})} \\
&= \underset{\mathbf{y}}{\operatorname{amax}} \exp\left\{\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\right\} \\
&= \underset{\mathbf{y}}{\operatorname{amax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)
\end{aligned}$$

- We can use the Viterbi algorithm

Making Predictions with Log-Linear Models

- For tractability, assume $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes into bigrams:

$$\mathbf{f}(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Given $\mathbf{w}$, given $\mathbf{x}_{1:n}$, find:

$$\begin{aligned} \underset{\mathbf{y}_{1:n}}{\text{argmax}} \Pr(\mathbf{y}_{1:n} | \mathbf{x}_{1:n}; \mathbf{w}) &= \underset{\mathbf{y}}{\text{amax}} \frac{\exp \left\{ \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\}}{Z(\mathbf{x}; \mathbf{w})} \\ &= \underset{\mathbf{y}}{\text{amax}} \exp \left\{ \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\} \\ &= \underset{\mathbf{y}}{\text{amax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

- We can use the Viterbi algorithm

Parameter Estimation in Log-Linear Models

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z(\mathbf{x}; \mathbf{w})}$$

- Given training data

$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\} \quad ,$$

- How to estimate $\mathbf{w}$?
- Define the conditional log-likelihood of the data:

$$L(\mathbf{w}) = \sum_{k=1}^{m} \log \Pr(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$$

- $L(\mathbf{w})$ measures how well $\mathbf{w}$ explains the data. A good value for $\mathbf{w}$ will give a high value for $\Pr(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$ for all $k = 1 \ldots m$.
- We want $\mathbf{w}$ that maximizes $L(\mathbf{w})$

Parameter Estimation in Log-Linear Models

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z(\mathbf{x}; \mathbf{w})}$$

- Given training data

$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\} \quad ,$$

- How to estimate $\mathbf{w}$?
- Define the conditional log-likelihood of the data:

$$L(\mathbf{w}) = \sum_{k=1}^{m} \log \Pr(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$$

- $L(\mathbf{w})$ measures how well $\mathbf{w}$ explains the data. A good value for $\mathbf{w}$ will give a high value for $\Pr(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$ for all $k = 1 \ldots m$.
- We want $\mathbf{w}$ that maximizes $L(\mathbf{w})$

Learning Log-Linear Models: Loss + Regularization

- Solve:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \overbrace{-L(\mathbf{w})}^{\text{Loss}} + \overbrace{\frac{\lambda}{2}||\mathbf{w}||^2}^{\text{Regularization}}$$

  where
  - The first term is the negative conditional log-likelihood
  - The second term is a regularization term, it penalizes solutions with large norm
  - $\lambda \in \mathbb{R}$ controls the trade-off between loss and regularization
- Convex optimization problem $\rightarrow$ gradient descent
- Two common losses based on log-likelihood that make learning tractable:
  - Local Loss (MEMM): assume that $\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$ decomposes
  - Global Loss (CRF): assume that $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes

Learning Log-Linear Models: Loss + Regularization

- Solve:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \overbrace{-L(\mathbf{w})}^{\text{Loss}} + \overbrace{\frac{\lambda}{2}||\mathbf{w}||^2}^{\text{Regularization}}$$

  where
  - The first term is the negative conditional log-likelihood
  - The second term is a regularization term, it penalizes solutions with large norm
  - $\lambda \in \mathbb{R}$ controls the trade-off between loss and regularization
- Convex optimization problem $\rightarrow$ gradient descent
- Two common losses based on log-likelihood that make learning tractable:
  - Local Loss (MEMM): assume that $\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$ decomposes
  - Global Loss (CRF): assume that $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes

Local Log-linear Loss (a.k.a. Maximum Entropy Markov Models)

McCallum, Freitag, and Pereira (2000)

- Similarly to HMMs:

$$
\begin{aligned}
\Pr(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}) &= \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \Pr(\mathbf{y}_{2:n} \mid \mathbf{x}_{1:n}, y_1) \\
&= \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} \Pr(y_i | \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) \\
&= \Pr(y_1 | \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} \Pr(y_i | \mathbf{x}_{1:n}, \mathbf{y}_{i-1})
\end{aligned}
$$

- Markov assumption under a local log-linear loss:

$$
\Pr(y_i | \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) = \Pr(y_i | \mathbf{x}_{1:n}, y_{i-1})
$$

Parameter Estimation with Local Log-Linear Markov Models

$$\Pr(y_{1:n} \mid \mathbf{x}_{1:n}) \quad = \quad \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} \Pr(y_i | \mathbf{x}_{1:n}, i, y_{i-1})$$

- The log-linear model is normalized locally (i.e. at each position):

$$\Pr(y \mid \mathbf{x}, i, y') = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y', y)\}}{Z(\mathbf{x}, i, y')}$$

- The log-likelihood is also local :

$$L(\mathbf{w}) = \sum_{k=1}^{m} \sum_{i=1}^{n^{(k)}} \log \Pr(\mathbf{y}_i^{(k)} | \mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}^{(k)})$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_j} = \frac{1}{m} \sum_{k=1}^{m} \sum_{i=1}^{n^{(k)}} \left[\overbrace{\mathbf{f}_j(\mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}^{(k)}, \mathbf{y}_i^{(k)})}^{\text{observed}} - \overbrace{\sum_{y \in \mathcal{Y}} \Pr(\mathbf{y} | \mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}^{(k)}, y) \; \mathbf{f}_j(\mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}^{(k)}, y)}^{\text{expected}} \right]$$

Conditional Random Fields

Lafferty, McCallum, and Pereira (2001)

- Log-linear model of the conditional distribution:

$$\Pr(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{\exp\{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})\}}{Z(\mathbf{x})}$$

  where
  - $\mathbf{x}$ and $\mathbf{y}$ are input and output sequences
  - $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a feature vector of $\mathbf{x}$ and $\mathbf{y}$ that decomposes into factors
  - $\mathbf{w}$ are model parameters
- To predict the best sequence

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \Pr(\mathbf{y}|\mathbf{x})$$

- Log-Likelihood at the global (sequence) level:

$$L(\mathbf{w}) = \sum_{k=1}^{m} \log \Pr(\mathbf{y}^{(k)}|\mathbf{x}^{(k)};\mathbf{w})$$

Computing the Gradient in CRFs

Consider a parameter $\mathbf{w}_j$ and its associated feature $\mathbf{f}_j$:

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_j} = \frac{1}{m} \sum_{k=1}^{m} \left[\overbrace{\mathbf{f}_j(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})}^{\text{observed}} - \overbrace{\sum_{\mathbf{y} \in \mathcal{Y}^*} \Pr(\mathbf{y} | \mathbf{x}^{(k)}; \mathbf{w}) \ \mathbf{f}_j(\mathbf{x}^{(k)}, \mathbf{y})}^{\text{expected}} \right]$$

where

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

- First term: observed value of $\mathbf{f}_j$ in training examples
- Second term: expected value of $\mathbf{f}_j$ under current $\mathbf{w}$
- In the optimal, observed = expected

Computing the Gradient in CRFs

- The first term is easy to compute, by counting explicitly

$$\sum_i \mathbf{f}_j(\mathbf{x}, i, y_{i-1}^{(k)}, y_i^{(k)})$$

- The second term is more involved,

$$\sum_{\mathbf{y} \in \mathcal{Y}^*} \Pr(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w}) \sum_i \mathbf{f}_j(\mathbf{x}^{(k)}, i, y_{i-1}, y_i)$$

  because it sums over all sequences $\mathbf{y} \in \mathcal{Y}^n$

- But there is an efficient solution ...

Computing the Gradient in CRFs

- For an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:

$$\sum_{\mathbf{y} \in \mathcal{Y}^n} \Pr(\mathbf{y} | \mathbf{x}^{(k)}; \mathbf{w}) \sum_{i=1}^{n} \mathbf{f}_j(\mathbf{x}^{(k)}, i, y_{i-1}, y_i) = \sum_{i=1}^{n} \sum_{a,b \in \mathcal{Y}} \mu_i^k(a, b) \mathbf{f}_j(\mathbf{x}^{(k)}, i, a, b)$$

- $\mu_i^k(a, b)$ is the marginal probability of having labels (a, b) at position i:

$$\mu_i^k(a, b) = \Pr(\langle i, a, b \rangle \mid \mathbf{x}^{(k)}; \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}^n \;:\; y_{i-1}=a,\; y_i=b} \Pr(\mathbf{y} | \mathbf{x}^{(k)}; \mathbf{w})$$

- The quantities μ_i^k can be computed efficiently in $O(nL^2)$ using the forward-backward algorithm

Forward-Backward for CRFs

- Assume fixed $\mathbf{x}$ and $\mathbf{w}$.
- For notational convenience, define the score of a label bigram as:

$$s(i,a,b) = \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x},i,a,b)\}$$

  such that we can write

$$\Pr(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x},\mathbf{y})\}}{Z(\mathbf{x})} = \frac{\exp\{\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x},i,y_{i-1},y_i)\}}{Z(\mathbf{x})} = \frac{\prod_{i=1}^{n} s(i,y_{i-1},y_i)}{Z}$$

- Normalizer: $Z = \sum_{\mathbf{y}} \prod_{i=1}^{n} s(i,y_{i-1},y_i)$
- Marginals: $\mu(i,a,b) = \frac{1}{Z} \sum_{\mathbf{y},\text{s.t.} y_{i-1}=a, y_i=b} \prod_{i=1}^{n} s(i,y_{i-1},y_i)$

Forward-Backward for CRFs

- **Definition:** forward and backward quantities

$$
\begin{aligned}
\alpha_i(a) &= \sum_{\mathbf{y}_{1:i} \in \mathcal{Y}^i : y_i = a} \prod_{j=1}^{i} s(j, y_{j-1}, y_j) \\
\beta_i(b) &= \sum_{\mathbf{y}_{i:n} \in \mathcal{Y}^{(n-i+1)} : y_i = b} \prod_{j=i+1}^{n} s(j, y_{j-1}, y_j)
\end{aligned}
$$

- $Z = \sum_a \alpha_n(a)$
- $\mu_i(a, b) = \{\alpha_{i-1}(a) * s(i, a, b)\} * \beta_i(b) * Z^{-1}\}$

- Similarly to Viterbi, $\alpha_i(a)$ and $\beta_i(b)$ can be computed recursively in $O(n|\mathcal{Y}|^2)$

Forward-Backward for CRFs

- **Definition:** forward and backward quantities

$$\begin{aligned} \alpha_i(a) &= \sum_{\mathbf{y}_{1:i} \in \mathcal{Y}^i : y_i = a} \prod_{j=1}^{i} s(j, y_{j-1}, y_j) \\ \beta_i(b) &= \sum_{\mathbf{y}_{i:n} \in \mathcal{Y}^{(n-i+1)} : y_i = b} \prod_{j=i+1}^{n} s(j, y_{j-1}, y_j) \end{aligned}$$

- $Z = \sum_a \alpha_n(a)$
- $\mu_i(a, b) = \{\alpha_{i-1}(a) * s(i, a, b)\} * \beta_i(b) * Z^{-1}\}$

- Similarly to Viterbi, $\alpha_i(a)$ and $\beta_i(b)$ can be computed recursively in $O(n|\mathcal{Y}|^2)$

CRFs: summary so far

- Log-linear models for sequence prediction, $\Pr(\mathbf{y}|\mathbf{x};\mathbf{w})$
- Computations factorize on label bigrams
- Model form:

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \sum_i \mathbf{w}\cdot\mathbf{f}(\mathbf{x},i,y_{i-1},y_i)$$

- Prediction: uses Viterbi (from HMMs)
- Parameter estimation:
  - Gradient-based methods, in practice L-BFGS
  - Computation of gradient uses forward-backward (from HMMs)

CRFs: summary so far

- Log-linear models for sequence prediction, $\Pr(\mathbf{y}|\mathbf{x};\mathbf{w})$
- Computations factorize on label bigrams
- Model form:

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Prediction: uses Viterbi (from HMMs)
- Parameter estimation:
  - Gradient-based methods, in practice L-BFGS
  - Computation of gradient uses forward-backward (from HMMs)

- Next Question: Local Loss or CRFs? HMMs or CRFs?

Local vs. Global Log-linear Losses

Local Loss: $\Pr(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\}}{Z(\mathbf{x}, i, y_{i-1}; \mathbf{w})}$

CRFs: $\Pr(\mathbf{y} \mid \mathbf{x}) = \frac{\exp\{\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\}}{Z(\mathbf{x})}$

- Both exploit the same factorization, i.e. same features
- Same computations to compute $\operatorname{argmax}_{\mathbf{y}} \Pr(\mathbf{y} \mid \mathbf{x})$
- Local loss is locally normalized; CRFs globally normalized
  - Local loss assumes that $\Pr(y_i \mid x_{1:n}, y_{1:i-1}) = \Pr(y_i \mid x_{1:n}, y_{i-1})$
  - Leads to "Label Bias Problem" (Lafferty et al., 2001; Andor et al., 2016)
- Local loss is cheaper to train (reduces to multiclass MaxEnt learning)
- CRFs are easier to extend to other structures

HMMs for sequence prediction

- $\mathbf{x}$ are the observations, $\mathbf{y}$ are the hidden states
- HMMs model the joint distributon $\Pr(\mathbf{x}, \mathbf{y})$
- Parameters: (assume $\mathcal{X} = \{1, \ldots, k\}$ and $\mathcal{Y} = \{1, \ldots, l\}$)
  - $\pi \in \mathbb{R}^l$, $\pi_a = \Pr(y_1 = a)$
  - $T \in \mathbb{R}^{l \times l}$, $T_{a,b} = \Pr(y_i = b | y_{i-1} = a)$
  - $O \in \mathbb{R}^{l \times k}$, $O_{a,c} = \Pr(x_i = c | y_i = a)$
- Model form

$$\Pr(\mathbf{x}, \mathbf{y}) = \pi_{y_1} O_{y_1, x_1} \prod_{i=2}^{n} T_{y_{i-1}, y_i} O_{y_i, x_i}$$

- Parameter Estimation: maximum likelihood by counting events and normalizing

HMMs and CRFs

- In CRFs: $\hat{\mathbf{y}} = \mathrm{amax}_{\mathbf{y}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$

- In HMMs:
  $\hat{\mathbf{y}} = \mathrm{amax}_{\mathbf{y}}\, \pi_{y_1} O_{y_1,x_1} \prod_{i=2}^{n} T_{y_{i-1},y_i} O_{y_i,x_i}$
  $= \mathrm{amax}_{\mathbf{y}} \log(\pi_{y_1} O_{y_1,x_1}) + \sum_{i=2}^{n} \log(T_{y_{i-1},y_i} O_{y_i,x_i})$

- An HMM can be expressed as factored linear models:

$\mathbf{f}_j(\mathbf{x}, i, y, y')$	$\mathbf{w}_j$
$i = 1$ & $y' = a$	$\log(\pi_a)$
$i > 1$ & $y = a$ & $y' = b$	$\log(T_{a,b})$
$y' = a$ & $x_i = c$	$\log(O_{a,b})$

- Hence, HMM are factored linear models

HMMs and CRFs: main differences

- Representation:
  - HMM “features” are tied to the generative process.
  - CRF features are **very** flexible. They can look at the whole input $\mathbf{x}$ paired with a label bigram (y_i, y_{i+1}).
  - In practice, for prediction tasks, “good” discriminative features can improve accuracy **a lot**.
- Parameter estimation:
  - HMMs focus on explaining the data, both $\mathbf{x}$ and $\mathbf{y}$.
  - CRFs focus on the mapping from $\mathbf{x}$ to $\mathbf{y}$.
  - A priori, it is hard to say which paradigm is better.
  - Same dilemma as Naive Bayes vs. Maximum Entropy.

Structured Prediction

Perceptron, SVMs, CRFs

Learning Structured Predictors

- Goal: given training data
$$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$$
learn a predictor $\mathbf{x} \rightarrow \mathbf{y}$ with small error on unseen inputs

- In a CRF:
$$\begin{aligned} \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^*} P(\mathbf{y}|\mathbf{x}; \mathbf{w}) &= \frac{\exp\left\{\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\right\}}{Z(\mathbf{x}; \mathbf{w})} \\ &= \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$
  - To predict new values, $Z(\mathbf{x}; \mathbf{w})$ is not relevant
  - Parameter estimation: $\mathbf{w}$ is set to maximize likelihood

- Can we learn $\mathbf{w}$ more directly, focusing on errors?

Learning Structured Predictors

- Goal: given training data
$$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$$
learn a predictor $\mathbf{x} \to \mathbf{y}$ with small error on unseen inputs

- In a CRF:
$$\begin{aligned} \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} P(\mathbf{y}|\mathbf{x};\mathbf{w}) &= \frac{\exp\left\{\sum_{i=1}^{n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)\right\}}{Z(\mathbf{x};\mathbf{w})} \\ &= \sum_{i=1}^{n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$
  - To predict new values, $Z(\mathbf{x};\mathbf{w})$ is not relevant
  - Parameter estimation: $\mathbf{w}$ is set to maximize likelihood

- Can we learn $\mathbf{w}$ more directly, focusing on errors?

The Structured Perceptron

Collins (2002)

- Set $\mathbf{w} = \mathbf{0}$
- For $t = 1 \ldots T$
  - For each training example $(\mathbf{x}, \mathbf{y})$
    1. Compute $\mathbf{z} = \mathrm{argmax}_{\mathbf{z}}\, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})$
    2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z})$$

- Return $\mathbf{w}$

The Structured Perceptron + Averaging

Freund and Schapire (1999); Collins (2002)

- Set $\mathbf{w} = \mathbf{0}$, $\mathbf{w}^a = \mathbf{0}$
- For $t = 1 \ldots T$
  - For each training example $(\mathbf{x}, \mathbf{y})$
    1. Compute $\mathbf{z} = \operatorname{argmax}_{\mathbf{z}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})$
    2. If $\mathbf{z} \neq \mathbf{y}$

       $$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z})$$

    3. $\mathbf{w}^{\mathbf{a}} = \mathbf{w}^{\mathbf{a}} + \mathbf{w}$
- Return $\mathbf{w}^{\mathbf{a}}/mT$, where m is the number of training examples

Perceptron Updates: Example

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{z}$	PER	LOC	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

- Let $\mathbf{y}$ be the correct output for $\mathbf{x}$.
- Say we predict $\mathbf{z}$ instead, under our current $\mathbf{w}$
- The update is:

$$\begin{aligned}
\mathbf{g} &= \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z}) \\
&= \sum_i \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) - \sum_i \mathbf{f}(\mathbf{x}, i, z_{i-1}, z_i) \\
&= \mathbf{f}(\mathbf{x}, 2, \text{PER}, \text{PER}) - \mathbf{f}(\mathbf{x}, 2, \text{PER}, \text{LOC}) \\
&+ \ \mathbf{f}(\mathbf{x}, 3, \text{PER}, \text{-}) - \mathbf{f}(\mathbf{x}, 3, \text{LOC}, \text{-})
\end{aligned}$$

- Perceptron updates are typically very sparse

Properties of the Perceptron

- Online algorithm. Often much more efficient than "batch" algorithms
- If the data is separable, it will converge to parameter values with 0 errors
- Number of errors before convergence is related to a definition of *margin*. Can also relate margin to generalization properties
- In practice:
  1. Averaging improves performance a lot
  2. Typically reaches a good solution after only a few (say 5) iterations over the training set
  3. Often performs nearly as well as CRFs, or SVMs

Averaged Perceptron Convergence

Iteration	Accuracy
1	90.79
2	91.20
3	91.32
4	91.47
5	91.58
6	91.78
7	91.76
8	91.82
9	91.88
10	91.91
11	91.92
12	91.96
...	

(results on validation set for a parsing task)

Margin-based Structured Prediction

- Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
- Model: $\mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$
- Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
  $\exists \mathbf{y} \neq \mathbf{y}^{(k)} \; : \; \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) < \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \Longrightarrow$ error
- Let $\mathbf{y}' = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}^* : \mathbf{y} \neq \mathbf{y}^{(k)}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})$
  Define $\gamma_k = \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}'))$
- The quantity γ_k is a notion of margin on example k:
  $\gamma_k > 0 \Longleftrightarrow$ no mistakes in the example
  high $\gamma_k \Longleftrightarrow$ high confidence

Margin-based Structured Prediction

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

- Model: $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$

- Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
  $\exists \mathbf{y} \neq \mathbf{y}^{(k)} \ : \ \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^{(k)},\mathbf{y}^{(k)}) < \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^{(k)},\mathbf{y}) \Longrightarrow$ error

- Let $\mathbf{y}' = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^*:\mathbf{y}\neq\mathbf{y}^{(k)}} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}^{(k)},\mathbf{y})$
  Define $\gamma_k = \mathbf{w}\cdot(\mathbf{f}(\mathbf{x}^{(k)},\mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)},\mathbf{y}'))$

- The quantity γ_k is a notion of margin on example k:
  $\gamma_k > 0 \Longleftrightarrow$ no mistakes in the example
  high $\gamma_k \Longleftrightarrow$ high confidence

Margin-based Structured Prediction

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

- Model: $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$

- Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
  $\exists \mathbf{y} \neq \mathbf{y}^{(k)} \; : \; \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) < \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \Longrightarrow$ error

- Let $\mathbf{y}' = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^* : \mathbf{y} \neq \mathbf{y}^{(k)}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})$
  Define $\gamma_k = \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}'))$

- The quantity γ_k is a notion of margin on example k:
  $\gamma_k > 0 \Longleftrightarrow$ no mistakes in the example
  high $\gamma_k \Longleftrightarrow$ high confidence

Structured Hinge Loss

Taskar, Guestrin, and Koller (2003)

- A margin-based loss function on example k:

$$L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) = \max_{\mathbf{y} \in \mathcal{Y}^*} \left\{ \overbrace{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})}^{\text{score of } \mathbf{y}} + \overbrace{e(\mathbf{y}^{(k)}, \mathbf{y})}^{\text{mistakes in } \mathbf{y}} - \overbrace{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})}^{\text{score of } \mathbf{y}^{(k)}} \right\}$$

- $e(\mathbf{y}^{(k)}, \mathbf{y})$ is a function counting the number of mistakes in $\mathbf{y}$ with respect to $\mathbf{y}^{(k)}$

  more mistakes in $\mathbf{y} \longrightarrow$ larger separation
- Leads to an SVM for structured prediction
- Given a training set, find:

$$\operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{k=1}^{m} L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Regularized Loss Minimization

- Given a training set $\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$.
  Find:

$$\underset{\mathbf{w} \in \mathbb{R}^D}{\operatorname{argmin}} \quad \sum_{k=1}^{m} L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Two common loss functions $L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
  - Log-likelihood loss (CRFs)

$$-\log P(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}; \mathbf{w})$$

  - Hinge loss (SVMs)

$$\max_{\mathbf{y} \in \mathcal{Y}^*} \left(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) + e(\mathbf{y}^{(k)}, \mathbf{y}) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \right)$$

Learning Structure Predictors: summary so far

- Linear models for sequence prediction

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Computations factorize on label bigrams
  - Decoding: using Viterbi
  - Marginals: using forward-backward
- Parameter estimation:
  - Perceptron, Log-likelihood, SVMs
  - Extensions from classification to the structured case
  - Optimization methods:
    - Stochastic (sub)gradient methods (LeCun et al., 1998; Shalev-Shwartz et al., 2011)
    - Exponentiated Gradient (Collins et al., 2008)
    - SVM Struct (Tsochantaridis et al., 2005)
    - Structured MIRA (Crammer et al., 2005)

Beyond Linear Sequence Prediction

Factored Sequence Prediction, Beyond Bigrams

- It is easy to extend the scope of features to k-grams

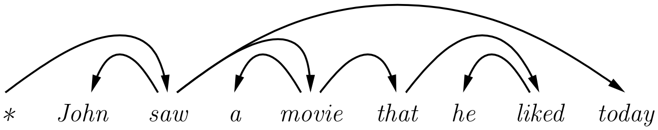
$$\mathbf{f}(\mathbf{x}, i, y_{i-k+1:i-1}, y_i)$$

- In general, think of state σ_i remembering relevant history
  - $\sigma_i = y_{i-1}$ for bigrams
  - $\sigma_i = y_{i-k+1:i-1}$ for k-grams
  - σ_i can be the state at time i of a deterministic automaton generating $\mathbf{y}$
- The structured predictor is

$$\underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \sigma_i, y_i)$$

- Viterbi and forward-backward extend naturally, in $O(nL^k)$

Dependency Structures



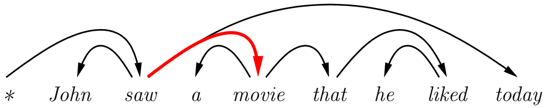
- Directed arcs represent dependencies between a head word and a modifier word.

- E.g.:

  movie *modifies* saw,
  John *modifies* saw,
  today *modifies* saw

Dependency Parsing: arc-factored models

McDonald, Pereira, Ribarov, and Hajič (2005)



- Parse trees decompose into single dependencies $\langle h, m \rangle$

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{\langle h,m \rangle \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$$

- Some features: $\mathbf{f}_1(\mathbf{x}, h, m) = [\ \text{"saw"} \ \rightarrow \text{"movie"}\ ]$
  $\mathbf{f}_2(\mathbf{x}, h, m) = [$ distance $= +2\ ]$
- Tractable inference algorithms exist

Linear Structured Prediction

- Sequence prediction (bigram factorization)

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Dependency parsing (arc-factored)

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{\langle h,m \rangle \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$$

- In general, we can enumerate parts $r \in \mathbf{y}$

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{r \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

Factored Sequence Prediction: from Linear to Non-linear

$$\mathrm{score}(\mathbf{x}, \mathbf{y}) = \sum_i \mathrm{s}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Linear:

$$\mathrm{s}(\mathbf{x}, i, y_{i-1}, y_i) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Non-linear, using a feed-forward neural network:

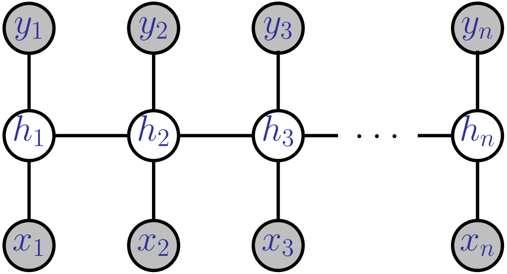
$$\mathrm{s}(\mathbf{x}, i, y_{i-1}, y_i) = \mathbf{w} \cdot [e_{y_{i-1}, y_i} \otimes h(\mathbf{f}(\mathbf{x}, i))]$$

  where:

$$h(\mathbf{f}(\mathbf{x}, i)) = \sigma(W^2 \sigma(W^1 \sigma(W^0 \mathbf{f}(\mathbf{x}, i))))$$

- Remarks:
  - The non-linear model computes a hidden representation of the input
  - Still factored: Viterbi and Forward-Backward work
  - Parameter estimation becomes non-convex, use backpropagation

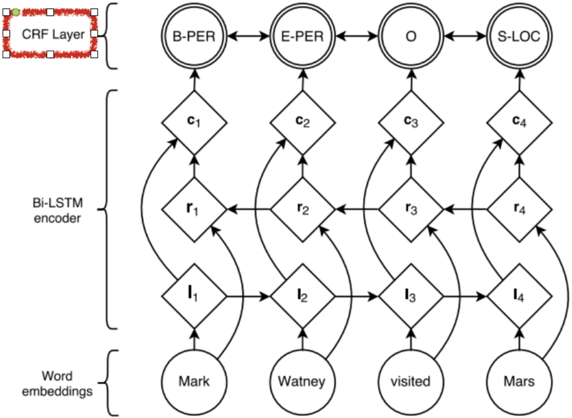
Recurrent Sequence Prediction



- Induction of hidden vectors (i.e. embeddings) that keep track of previous observations and predictions
- Making predictions is not tractable
  - In practice: greedy predictions or beam search
- Learning is non-convex, so what?
- Popular methods: RNN, LSTM, Spectral Models, . . .

Neural Architectures for Named Entity Recognition

Guillaume Lample♠ Miguel Ballesteros♣♠
Sandeep Subramanian♠ Kazuya Kawakami♠ Chris Dyer♠

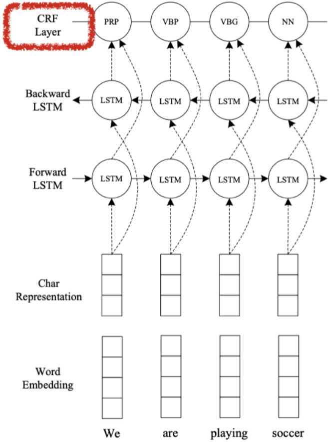


Model	**F$_1$**
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	**91.2**
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**
S-LSTM (no char)	87.96
S-LSTM	90.33

Table 1: English NER results (CoNLL-2003 test set).

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF

Xuezhe Ma and **Eduard Hovy**



	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

Table 3: Performance of our model on both the development and test sets of the two tasks, together with three baseline systems.

Model	**Acc.**
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

Table 4: POS tagging accuracy of our model on test data from WSJ proportion of PTB, together

Model	**F1**
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)‡	89.59
Huang et al. (2015)‡	90.10
Chiu and Nichols (2015)‡	90.77
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Passos et al. (2014)	90.90
Lample et al. (2016)‡	90.94
Luo et al. (2015)	91.20
This paper	**91.21**

Table 5: NER F1 score of our model on test data set from CoNLL-2003. For the purpose of com-

Thanks!

References I

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1231.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822, 2008.

Koby Crammer, Ryan McDonald, and Fernando Pereira. Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*, 2005.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, December 1999. ISSN 0885-6125. doi: 10.1023/A:1007662407062. URL http://dx.doi.org/10.1023/A:1007662407062.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968. Association for Computational Linguistics, 2006.

Sanjiv Kumar and Martial Hebert. Man-made structure detection in natural images using a causal multiscale random field. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, pages 119–126, 2003. doi: 10.1109/CVPR.2003.1211345. URL https://doi.org/10.1109/CVPR.2003.1211345.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

References II

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in neural information processing systems*, volume 16, 2003.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484, 2005.