Natural Language Processing (CSE 517): Sequence Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

April 25, 2018

Where We Are

- Language models
- Text classification
- **Linguistic analysis**
- Generation

Linguistic Analysis: Overview

Every linguistic analyzer is comprised of:

1. Theoretical motivation from linguistics and/or the text domain
2. An algorithm that maps $\mathcal{V}^\dagger$ to some output space $\mathcal{Y}$.
   - In this class, I'll start with abstract algorithms applicable to many problems.
3. An implementation of the algorithm
   - Once upon a time: rule systems and crafted rules
   - Most common now: supervised learning from annotated data
   - Frontier: less supervision (semi-, un-, distant, . . .)

Sequence Labeling

After text classification ($\mathcal{V}^\dagger \rightarrow \mathcal{L}$), the next simplest type of output is a **sequence labeling**.

$$\langle x_1, x_2, \ldots, x_\ell \rangle \mapsto \langle y_1, y_2, \ldots, y_\ell \rangle$$

Every word (or character) gets a label in $\mathcal{L}$.
Example problems:

- part-of-speech tagging (Church, 1988)
- spelling correction (Kernighan et al., 1990)
- word alignment (Vogel et al., 1996)
- named-entity recognition (Bikel et al., 1999)
- compression (Conroy and O'Leary, 2001)

The Simplest Sequence Labeler

Define features of a labeled word in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\begin{aligned} \hat{y}_i &= \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y) \\ &\overset{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y) \end{aligned}$$

The Simplest Sequence Labeler

Define features of a labeled word in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\begin{aligned}\hat{y}_i &= \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y) \\ &\overset{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y)\end{aligned}$$

Sometimes this works!

The Simplest Sequence Labeler

Define features of a labeled word in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\begin{aligned} \hat{y}_i &= \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y) \\ &\overset{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y) \end{aligned}$$

Sometimes this works!

We can do better when there are predictable relationships between Y_i and Y_{i+1}.

Generative Sequence Labeling: Hidden Markov Models

$$p(\boldsymbol{x}, \boldsymbol{y}) = \pi_{y_0} \prod_{i=1}^{\ell+1} \theta_{x_i | y_i} \cdot \gamma_{y_i | y_{i-1}}$$

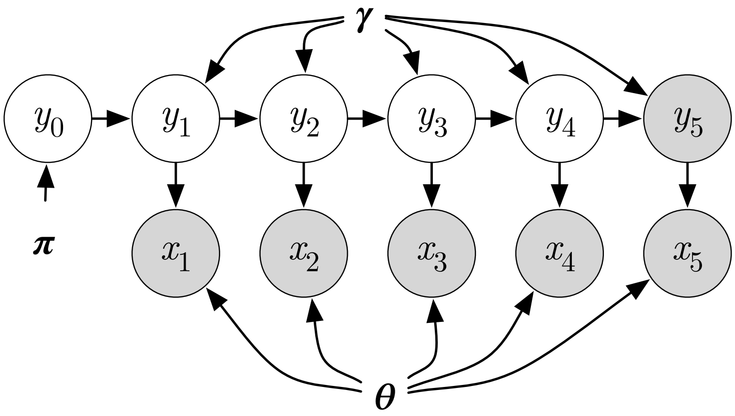
For each state/label $y \in \mathcal{L}$:

- $\boldsymbol{\theta}_{*|y}$ is the "emission" distribution
- $\boldsymbol{\gamma}_{*|y}$ is called the "transition" distribution

We saw this model before (Brown clustering). Differences:

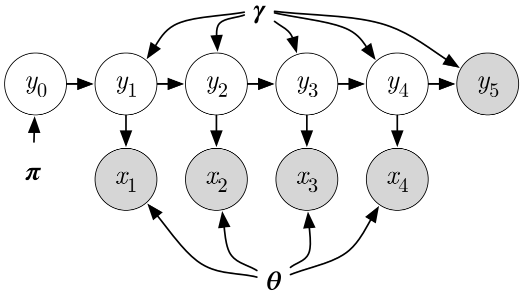
- We used "$\boldsymbol{z}$" before, now it's "$\boldsymbol{y}$"
- Before, we wanted to *discover* each y_i ("unsupervised")
- Now, we want to map $\boldsymbol{x} \mapsto \boldsymbol{y}$, defined within a task (might be supervised or not)

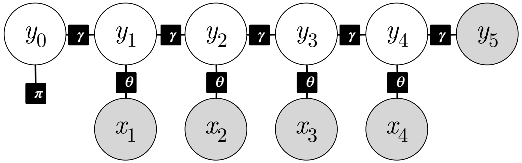
Graphical Reprsentation of Hidden Markov Models



Note: handling of beginning and end of sequence is a bit different than before. From here on, ignore last x since $\theta_{\bigcirc|\bigcirc} = 1$.

Factor Graph Representation of Hidden Markov Models





A More General Form

Twice now, we've made the move from generative models based on repeated "rolls of dice" to discriminative models based on feature representations.

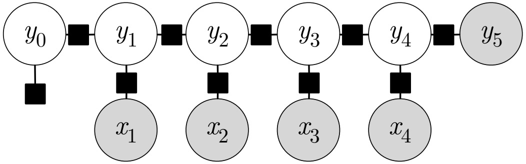
- Language modeling
- Text classification

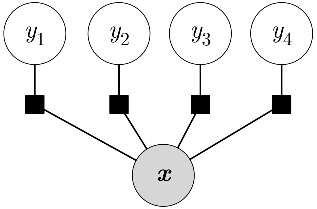
In the structured case, we can do the same thing.

$$\begin{aligned}
&\operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} p(y_0) \prod_{i=1}^{\ell+1} p(x_i, y_i \mid y_{i-1}) \\
&= \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} \log p(y_0) + \sum_{i=1}^{\ell+1} \log p(x_i, y_i \mid y_{i-1}) \\
&= \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} \sum_{i=1}^{\ell+1} \mathbf{w} \cdot \boldsymbol{\phi}(x_i, y_i, y_{i-1})
\end{aligned}$$

In this case, each Y_i "interacts" with Y_{i-1} and Y_{i+1} directly.

Structured vs. Not





Each of these has an advantage over the other:

- The HMM lets the different labels “interact.”
- The simple unstructured classifier makes all of $\boldsymbol{x}$ available for every decision.

A More Powerful Solution

Slightly more generally, define features of adjacent labels in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$.

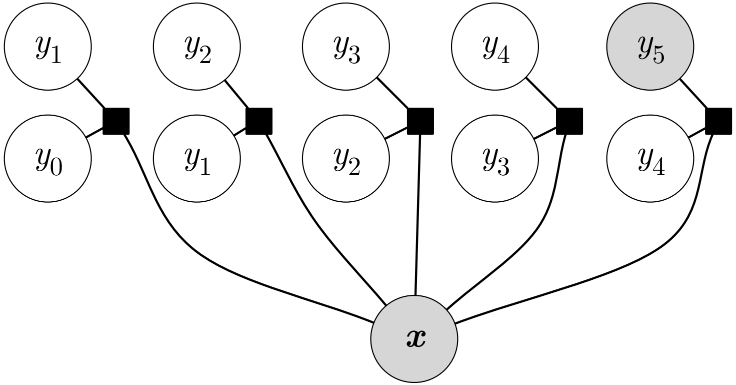
Features can depend on *any words at all*; this turns out not to affect asymptotic cost of prediction!

Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y, y' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$$

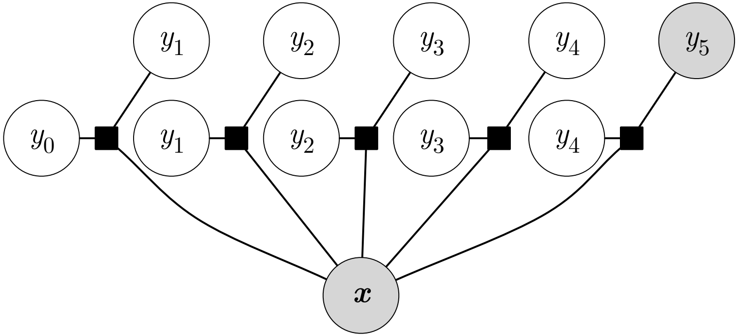
Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y, y' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$$



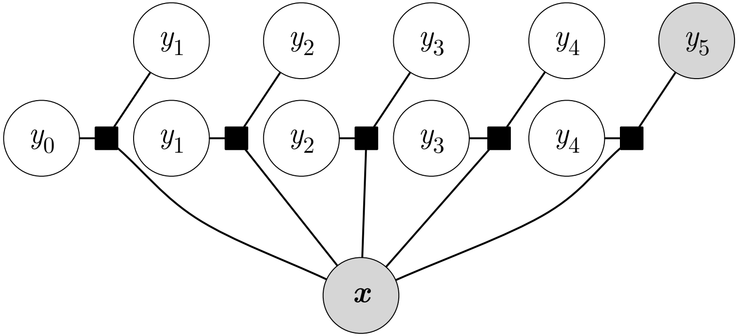
Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y,y' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$$



Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y, y' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$$



The problem is with disagreements: what if the $Y_{1:2}$ prediction and the $Y_{2:3}$ prediction do not agree about Y_2?

Even More Powerful: “Global” Prediction

As with the pairwise model, define features of adjacent labeled words in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$

“Structured” classifer/predictor:

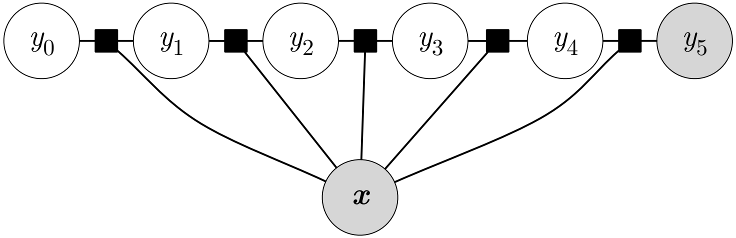
$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} \sum_{i=1}^{\ell+1} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y_i, y_{i-1})$$

Even More Powerful: "Global" Prediction

As with the pairwise model, define features of adjacent labeled words in context: $\boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$

"Structured" classifer/predictor:

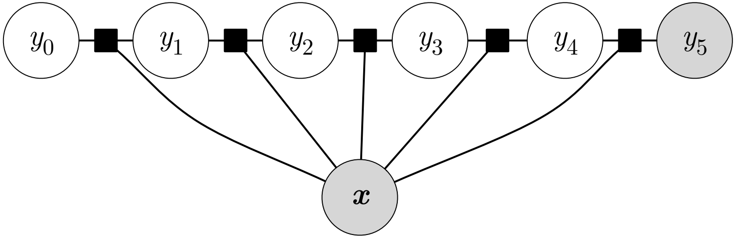
$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} \sum_{i=1}^{\ell+1} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y_i, y_{i-1})$$



Even More Powerful: "Global" Prediction

As with the pairwise model, define features of adjacent labeled words in context:
$\boldsymbol{\phi}(\boldsymbol{x}, i, y, y')$
"Structured" classifer/predictor:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} \sum_{i=1}^{\ell+1} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y_i, y_{i-1})$$



This is a fundamentally different kind of problem, demanding new:

- predicting ("decoding") algorithms
- training algorithms (to be discussed later)

Prediction with HMMs

We'll start with the classical HMM, then return later to the featurized case.

$$\operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} p(y_0) \prod_{i=1}^{\ell+1} p(x_i, y_i \mid y_{i-1})$$

How to optimize over $|\mathcal{L}|^{\ell}$ choices without explicit enumeration?

Prediction with HMMs

We'll start with the classical HMM, then return later to the featurized case.

$$\operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell+1}} p(y_0) \prod_{i=1}^{\ell+1} p(x_i, y_i \mid y_{i-1})$$

How to optimize over $|\mathcal{L}|^\ell$ choices without explicit enumeration?

Key: exploit the conditional independence assumptions:

$$Y_i \perp \boldsymbol{Y}_{1:i-2} \mid Y_{i-1}$$
$$Y_i \perp \boldsymbol{Y}_{i+2:\ell} \mid Y_{i+1}$$

Part-of-Speech Tagging Example

	I	suspect	the	present	forecast	is	pessimistic	.
noun	●	●	●	●	●	●		
adj.		●		●	●		●	
adv.				●				
verb		●		●	●	●		
num.	●							
det.			●					
punc.								●

With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

Two Obvious Solutions

Brute force: Enumerate all solutions, score them, pick the best.

Greedy: Pick each $\hat{y}_i$ according to:

$$\hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} p(y \mid \hat{y}_{i-1}) \cdot p(x_i \mid y)$$

What's wrong with these?

Conditional Independence

We can get an exact solution in polynomial time!

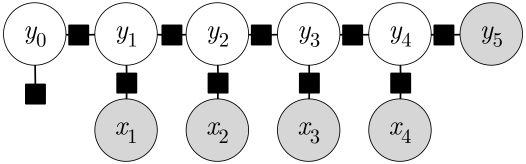
$$Y_i \perp \boldsymbol{Y}_{1:i-2} \mid Y_{i-1}$$
$$Y_i \perp \boldsymbol{Y}_{i+2:\ell} \mid Y_{i+1}$$

Given the adjacent labels to Y_i, others do not matter.

Let's start at the last position, ℓ . . .

The End of the Sequence

	x_1	x_2	$\ldots$	x_ℓ
y				
y'				
$\vdots$				
y^{last}				



$$
\begin{aligned}
p(Y_\ell = y \mid \boldsymbol{x}, \boldsymbol{y}_{1:(\ell-1)}) &= p(Y_\ell = y \mid X_\ell = x_\ell, Y_{\ell-1} = y_{\ell-1}, Y_{\ell+1} = \text{⯃}') \\
&= \gamma_{\text{⯃}|y} \cdot \theta_{x_\ell|y} \cdot \gamma_{y|y_{\ell-1}}
\end{aligned}
$$

The decision about Y_ℓ is a function of $y_{\ell-1}$, $\boldsymbol{x}$, and nothing else!

High-Level View of Viterbi

- The decision about Y_ℓ is a function of $y_{\ell-1}$, $\boldsymbol{x}$, and nothing else!

High-Level View of Viterbi

- The decision about Y_ℓ is a function of $y_{\ell-1}$, $\boldsymbol{x}$, and nothing else!
- If, for each value of $y_{\ell-1}$, we knew the best $\boldsymbol{y}_{1:(\ell-1)}$, then picking y_ℓ (and $y_{\ell-1}$) would be easy.

High-Level View of Viterbi

- The decision about Y_ℓ is a function of $y_{\ell-1}$, $\boldsymbol{x}$, and nothing else!
- If, for each value of $y_{\ell-1}$, we knew the best $\boldsymbol{y}_{1:(\ell-1)}$, then picking y_ℓ (and $y_{\ell-1}$) would be easy.
- Idea: for each position i, calculate the score of the best label prefix $\boldsymbol{y}_{1:i}$ ending in each possible value for Y_i.

High-Level View of Viterbi

- The decision about Y_ℓ is a function of $y_{\ell-1}$, $\boldsymbol{x}$, and nothing else!
- If, for each value of $y_{\ell-1}$, we knew the best $\boldsymbol{y}_{1:(\ell-1)}$, then picking y_ℓ (and $y_{\ell-1}$) would be easy.
- Idea: for each position i, calculate the score of the best label prefix $\boldsymbol{y}_{1:i}$ ending in each possible value for Y_i.
- With a little bookkeeping, we can then trace backwards and recover the best label sequence.

Recurrence

First, think about the *score* of the best sequence.

Let $s_i(y)$ be the score of the best label sequence for $\boldsymbol{x}_{1:i}$ that ends in y. It is defined recursively:

$$s_\ell(y) = \gamma_{\text{🛑}|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')}$$

Recurrence

First, think about the *score* of the best sequence.

Let $s_i(y)$ be the score of the best label sequence for $\boldsymbol{x}_{1:i}$ that ends in y. It is defined recursively:

$$s_\ell(y) = \gamma_{\text{⯃}|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')}$$

$$s_{\ell-1}(y) = \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-2}(y')}$$

Recurrence

First, think about the *score* of the best sequence.

Let $s_i(y)$ be the score of the best label sequence for $\boldsymbol{x}_{1:i}$ that ends in y. It is defined recursively:

$$\begin{aligned}
s_\ell(y) &= \gamma_{\bigcirc|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')} \\
s_{\ell-1}(y) &= \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-2}(y')} \\
s_{\ell-2}(y) &= \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-3}(y')}
\end{aligned}$$

Recurrence

First, think about the *score* of the best sequence.

Let $s_i(y)$ be the score of the best label sequence for $\boldsymbol{x}_{1:i}$ that ends in y. It is defined recursively:

$$
\begin{aligned}
s_\ell(y) &= \gamma_{\text{⯃}|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')} \\
s_{\ell-1}(y) &= \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-2}(y')} \\
s_{\ell-2}(y) &= \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-3}(y')} \\
&\vdots \\
s_i(y) &= \theta_{x_i|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{i-1}(y')}
\end{aligned}
$$

Recurrence

First, think about the *score* of the best sequence.
Let $s_i(y)$ be the score of the best label sequence for $\boldsymbol{x}_{1:i}$ that ends in y. It is defined recursively:

$$s_\ell(y) = \gamma_{\bigcirc|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')}$$

$$s_{\ell-1}(y) = \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-2}(y')}$$

$$s_{\ell-2}(y) = \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-3}(y')}$$

$$\vdots$$

$$s_i(y) = \theta_{x_i|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{i-1}(y')}$$

$$\vdots$$

$$s_1(y) = \theta_{x_1|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \pi_{y'}$$

Viterbi Procedure (Part I: Prefix Scores)

	x_1	x_2	$\ldots$	x_ℓ
y				
y'				
$\vdots$				
y^{last}				

Viterbi Procedure (Part I: Prefix Scores)

	x_1	x_2	$\dots$	x_ℓ
y	$s_1(y)$			
y'	$s_1(y')$			
$\vdots$				
y^{last}	$s_1(y^{last})$			

$$s_1(y) = \theta_{x_1|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \pi_{y'}$$

Viterbi Procedure (Part I: Prefix Scores)

	x_1	x_2	$\ldots$	x_ℓ
y	$s_1(y)$	$s_2(y)$		
y'	$s_1(y')$	$s_2(y')$		
$\vdots$				
y^{last}	$s_1(y^{last})$	$s_2(y^{last})$		

$$s_i(y) = \theta_{x_i|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{i-1}(y')}$$

Viterbi Procedure (Part I: Prefix Scores)

	x_1	x_2	$\ldots$	x_ℓ
y	$s_1(y)$	$s_2(y)$		$s_\ell(y)$
y'	$s_1(y')$	$s_2(y')$		$s_\ell(y')$
$\vdots$				
y^{last}	$s_1(y^{last})$	$s_2(y^{last})$		$s_\ell(y^{last})$

$$s_\ell(y) = \gamma_{\bigcirc|y} \cdot \theta_{x_\ell|y} \cdot \max_{y' \in \mathcal{L}} \gamma_{y|y'} \cdot \boxed{s_{\ell-1}(y')}$$

References I

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1–3):211–231, 1999.

Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of ANLP*, 1988.

John M. Conroy and Dianne P. O'Leary. Text summarization via hidden Markov models. In *Proc. of SIGIR*, 2001.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proc. of COLING*, 1990.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proc. of COLING*, 1996.

References I

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1–3):211–231, 1999.

Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of ANLP*, 1988.

John M. Conroy and Dianne P. O'Leary. Text summarization via hidden Markov models. In *Proc. of SIGIR*, 2001.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proc. of COLING*, 1990.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proc. of COLING*, 1996.