# Latent-variable CRF for SMT

Wilker Aziz

Universiteit van Amsterdam
w.aziz@uva.nl

May 9, 2017

# Conditional random field

Lafferty et al. [2001]

$$P(y|x, w) = \frac{\exp(w^\top \phi(x, y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(w^\top \phi(x, y'))} \tag{1}$$

- $\phi$ is a feature function mapping $(x, y)$ to $\mathbb{R}^d$
- $w$ is a feature vector in $\mathbb{R}^d$
- $Z(x|w) = \sum_{y \in \mathcal{Y}(x)} \exp(w^\top \phi(x, y))$ must be finite

# Flexible (overlapping) features

Examples

**As**$_1$ menin**as**$_2$ for**am**$_3$ pra$_4$ lá$_4$ $\leftrightarrow$ The$_1$ girls$_2$ went$_3$ over$_4$ there$_4$

Apague$_1$ a$_2$ luz$_3$ $\leftrightarrow$ Switch$_1$ the$_2$ light$_3$ off$_1$

Hard to account for with directed models
(due to causality assumptions)

# Global normalisation

Local models are trained with positive context only

- MLE: maximise the likelihood of observation $(c, o)$ under $P(O|C = c)$

# Global normalisation

Local models are trained with positive context only

- MLE: maximise the likelihood of observation $(c, o)$ under $P(O|C = c)$
- Label bias [Lafferty et al., 2001]

# Global normalisation

Local models are trained with positive context only

- MLE: maximise the likelihood of observation $(c, o)$ under $P(O|C = c)$
- Label bias [Lafferty et al., 2001]

What happens when we query the model with predicted contexts?

# Maximum likelihood estimation

Likelihood of an observation $(x, y)$

$$\mathcal{L}(w|x,y) = \log P(y|x,w) \tag{2}$$
$$= w^\top \phi(x,y) - \log Z(x|w) \tag{3}$$

# Maximum likelihood estimation

Likelihood of an observation $(x, y)$

$$\begin{aligned} \mathcal{L}(w|x, y) &= \log P(y|x, w) && (2) \\ &= w^\top \phi(x, y) - \log Z(x|w) && (3) \end{aligned}$$

Gradient-based optimisation

$$\nabla_w \mathcal{L}(w|x, y) = \phi(x, y) - \mathbb{E}_{P(Y|X=x, w)}[\phi(X, Y)] \qquad (4)$$

# Maximum likelihood estimation

Likelihood of an observation $(x, y)$

$$\mathcal{L}(w|x, y) = \log P(y|x, w) \tag{2}$$
$$= w^\top \phi(x, y) - \log Z(x|w) \tag{3}$$

Gradient-based optimisation

$$\nabla_w \mathcal{L}(w|x, y) = \phi(x, y) - \mathbb{E}_{P(Y|X=x,w)}[\phi(X, Y)] \tag{4}$$

Expected features should match the features of the observation

# LV-CRF

Model

$$P(y, d|x) = \frac{\exp(w^\top \phi(x, y, d))}{\sum_{y' \in \mathcal{Y}(x)} \sum_{d' \in \mathcal{D}(x,y')} \exp(w^\top \phi(x, y', d'))} \tag{5}$$

- $d$ is latent
- $Z(x, y|w) = \sum_{d \in \mathcal{D}(x,y)} \exp(w^\top \phi(x, y, d))$ must be finite

# MLE for LV-CRF

Likelihood of an observation $(x, y)$

$$\mathcal{L}(w|x,y) = \log P(y|x,w) \tag{6}$$

$$= \log \sum_{d \in \mathcal{D}(x,y)} P(y,d|x,w) \tag{7}$$

# MLE for LV-CRF

Likelihood of an observation $(x, y)$

$$\mathcal{L}(w|x, y) = \log P(y|x, w) \tag{6}$$

$$= \log \sum_{d \in \mathcal{D}(x,y)} P(y, d|x, w) \tag{7}$$

Gradient-based optimisation [Mann and McCallum, 2007]

$$\nabla_w \mathcal{L}(w|x, y) = \mathbb{E}_{P(D|X=x, Y=y, w)}[\phi(X, Y, D)] \tag{8}$$

$$- \mathbb{E}_{P(Y,D|X=x, w)}[\phi(X, Y, D)] \tag{9}$$

# MLE for LV-CRF

Likelihood of an observation $(x, y)$

$$\begin{aligned} \mathcal{L}(w|x, y) &= \log P(y|x, w) && (6) \\ &= \log \sum_{d \in \mathcal{D}(x,y)} P(y, d|x, w) && (7) \end{aligned}$$

Gradient-based optimisation [Mann and McCallum, 2007]

$$\begin{aligned} \nabla_w \mathcal{L}(w|x, y) &= \mathbb{E}_{P(D|X=x, Y=y, w)}[\phi(X, Y, D)] && (8) \\ &- \mathbb{E}_{P(Y, D|X=x, w)}[\phi(X, Y, D)] && (9) \end{aligned}$$

Expected features should match the features of the expected observation

# Note on training

Undirected models are considerably harder to learn

- expensive global normalisation
- complex joint distributions

# Note on training

Undirected models are considerably harder to learn

- expensive global normalisation
- complex joint distributions

Particularly hard with latent variables

- Alignment: Dyer et al. [2011]
- SMT: Blunsom et al. [2008] and Blunsom and Osborne [2008]

# Note on training

Undirected models are considerably harder to learn

- expensive global normalisation
- complex joint distributions

Particularly hard with latent variables

- Alignment: Dyer et al. [2011]
- SMT: Blunsom et al. [2008] and Blunsom and Osborne [2008]
- Approximate techniques
    - Contrastive divergence [Hinton, 2002]
    - Contrastive estimation [Smith and Eisner, 2005]
    - Piecewise training [Sutton and McCallum, 2005]

# SMT with CRFs

Blunsom et al. [2008]

- $d$ is a derivation complying with a *hiero* grammar
- $\phi$ featurises steps in a synchronous derivation

$$P(y, d|x) = \frac{\exp\left(\sum_{r_{s,t} \in d} w^\top \phi(r_{s,t}|x, y, d)\right)}{\sum_{d' \in \mathcal{D}(x)} \exp\left(\sum_{r_{s,t} \in d} w^\top \phi(r_{s,t}|x, y', d')\right)}$$

- $\mathcal{D}(x)$ is the space of derivations over target strings aligned to the source string $x$
- in the denominator $y'$ is defined implicitly as yield$(d')$
- $r_{s,t}$ is a synchronous rule decorated with a source span $s$ and a target span $t$

# ITG with CRFs

In Project 2, you will use an ITG

$$
\begin{array}{lcl}
S & \rightarrow & X \\
X & \rightarrow & X\ X \\
X & \rightarrow & x/y \text{ for all } x \in \Sigma \text{ and } y \in \Delta \\
X & \rightarrow & \epsilon/y \text{ for all } y \in \Delta \\
X & \rightarrow & x/\epsilon \text{ for all } x \in \Sigma
\end{array}
$$

# Global normalisation

ITGs are capable of unbounded insertion (hiero grammars are not!)

# Global normalisation

ITGs are capable of unbounded insertion (hiero grammars are not!)

Normaliser may diverge for certain $w \in \mathbb{R}^d$

- $\mathcal{D}(x)$ is typically inifinite
- $\sum_{d \in \mathcal{D}(x)} \exp\left(\sum_{r_{s,t} \in d} w^\top \phi(r_{s,t} | x, y, d)\right)$

# Global normalisation

ITGs are capable of unbounded insertion (hiero grammars are not!)

Normaliser may diverge for certain $w \in \mathbb{R}^d$

- $\mathcal{D}(x)$ is typically inifinite
- $$\sum_{d \in \mathcal{D}(x)} \exp\left(\sum_{r_{s,t} \in d} w^\top \phi(r_{s,t} | x, y, d)\right)$$

Solution: constrain strings by length

- introduce a constrain $n$ that depends on $x$
- make $\mathcal{D}_n(x)$ such that $|\text{yield}(d)| \leq n$ for $d \in \mathcal{D}_n(x)$
- $$\sum_{d \in \mathcal{D}_n(x)} \exp\left(\sum_{r_{s,t} \in d} w^\top \phi(r_{s,t} | x, y, d)\right)$$

# Learning

Likelihood of an observation $(x, y, n)$

$$\mathcal{L}(w|x,y,n) = \log P(y|x,n,w) \tag{10}$$

$$= \log \sum_{d \in \mathcal{D}(x,y)} P(y,d|x,n,w) \tag{11}$$

note that $\mathcal{D}(x, y)$ is always finite

# Learning

Likelihood of an observation $(x, y, n)$

$$\mathcal{L}(w|x, y, n) = \log P(y|x, n, w) \quad (10)$$

$$= \log \sum_{d \in \mathcal{D}(x,y)} P(y, d|x, n, w) \quad (11)$$

note that $\mathcal{D}(x, y)$ is always finite

Gradient-based optimisation

$$\nabla_w \mathcal{L}(w|x, y, n) = \underbrace{\mathbb{E}_{P(D|X=x, Y=y, n, w)}[\phi(X, Y, D)]}_{\text{expected features for observation } (x,y)} \quad (12)$$

$$- \underbrace{\mathbb{E}_{P(Y, D|X=x, n, w)}[\phi(X, Y, D)]}_{\text{expected features for observation } x} \quad (13)$$

# What do we need?

An ITG parser in order to obtain

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$
   finite set of derivations over target strings that align to the source string $x$ where the target string is no longer than $n$ words

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$
   finite set of derivations over target strings that align to the source string $x$ where the target string is no longer than $n$ words
3. $\mathcal{D}(x, y)$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$
   finite set of derivations over target strings that align to the source string $x$ where the target string is no longer than $n$ words
3. $\mathcal{D}(x, y)$
   set of derivations of the string pair $(x, y)$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$
   finite set of derivations over target strings that align to the source string $x$ where the target string is no longer than $n$ words
3. $\mathcal{D}(x, y)$
   set of derivations of the string pair $(x, y)$
4. expected feature vector of observations $(x, y)$

# What do we need?

An ITG parser in order to obtain

1. $\mathcal{D}(x)$
   (potentially infinite) set of derivations over target strings that align to the source string $x$
2. $\mathcal{D}_n(x)$
   finite set of derivations over target strings that align to the source string $x$ where the target string is no longer than $n$ words
3. $\mathcal{D}(x, y)$
   set of derivations of the string pair $(x, y)$
4. expected feature vector of observations $(x, y)$
5. expected feature vector of an observation $x$

# Notation

Using a hypergraph view

- $u, v, s$ are nodes
- $e$ is an edge
- $\mathrm{head}(e)$ is a node
- $\mathrm{tail}(e)$ is a sequence of nodes
- if $u \in \mathrm{tail}(e)$ and $v \in \mathrm{tail}(e)$, $u$ and $v$ are siblings
- Backward-star: $BS(u)$ is the set of edges incoming to $u$
  $u$ is the head of the edge
- Forward-star: $FS(u)$ is the set of edges outgoing from $u$
  $u$ is in the tail of the edge
- $w(e)$ is the weight of an edge

# Inside

The INSIDE recursion

$$I(v) = \begin{cases} \bar{1} & \text{if } BS(v) = \emptyset \\ \bigoplus_{e \in BS(v)} w(e) \bigotimes_{u \in \text{tail}(e)} \beta(u) & \text{otherwise} \end{cases} \tag{14}$$

For acyclic hypergraphs

# Outside

The OUTSIDE recursion

$$O(v) = \begin{cases} \bar{1} & \text{if } FS(v) = \emptyset \\ \bigoplus_{e \in FS(v)} O(\text{head}(e)) \bigotimes_{s \in tail(e) \setminus \{v\}} I(s) & \text{otherwise} \end{cases} \quad (15)$$

For acyclic hypergraphs

# Topsort

**function** $\textsc{TopSort}(G = \langle V, \langle E, w \rangle\rangle)$
    $S = \{v \in V : BS(v) = \emptyset\}$ ▷ nodes with no dependencies
    $D = \{v \mapsto \{u : \exists e \in BS(v) \wedge u \in \mathrm{tail}(e)\} : v \in V\}$
    ▷ a node depends on all of its children
    $L = \langle\rangle$ ▷ top-sorted nodes
    **while** $S \neq \emptyset$ **do**
        $u \leftarrow$ pop$(S)$ ▷ remove and return a node from $S$
        $L \leftarrow L + \langle u \rangle$ ▷ append $u$ to $L$
        **for** $e$ in $FS(u)$ **do** ▷ outgoing edges from $u$
            $v \leftarrow \mathrm{head}(e)$ ▷ parent of $u$ in $e$
            $D(v) \leftarrow D(v) \setminus \{u\}$ ▷ remove $u$ from $D(v)$
            **if** $D(v) == \emptyset$ **then** ▷ $v$'s dependencies have been sorted
                $S \leftarrow S \cup \{v\}$
            **end if**
        **end for**
    **end while**
    **return** $L$
**end function**

# Inside

```
function Inside(G = ⟨V, ⟨E, w⟩⟩)
    for v in TopSort(G) do                                  ▷ visit nodes bottom-up
        if BS(v) == ∅ then
            I[v] ← 1̄                                        ▷ leaves
        else
            I[v] ← 0̄
            for e ∈ BS(v) do
                k ← w(e)                                    ▷ include the edge's own weight
                for u in tail(e) do
                    k ← k ⊗ I[u]
                end for
                I[v] ← I[v] ⊕ k                             ▷ accumulate for each edge
            end for
        end if
    end for
    return I
end function
```

# Outside

**function** $\textsc{Outside}(G = \langle V, \langle E, w \rangle \rangle, I, \text{root})$<br>
    $O[v] \leftarrow \bar{0}$ **for** $v \in V$<br>
    $O[\text{root}] \leftarrow \bar{1}$ ▷ this is the goal node<br>
    **for** $v$ in $\textsc{reverse}(\textsc{TopSort}(G))$ **do** ▷ visit nodes top-down<br>
        **for** $e \in BS(v)$ **do** ▷ $q$'s incoming edges<br>
            **for** $u \in \text{tail}(e)$ **do** ▷ children of $v$ in $e$<br>
                $k \leftarrow w(e) \otimes O[v]$<br>
                **for** $s$ in $\text{tail}(e)$ **do** ▷ siblings of $u$ in $e$<br>
                    **if** $u \neq s$ **then** ▷ $u$ itself is excluded<br>
                        $k \leftarrow k \otimes I[s]$<br>
                    **end if**<br>
                **end for**<br>
                $O[u] \leftarrow O[u] \oplus k$ ▷ accumulate it for $u$<br>
            **end for**<br>
        **end for**<br>
    **end for**<br>
    **return** $O$<br>
**end function**

# Expected features

```
function ExpectedFeatures(G = ⟨V, ⟨E, w⟩⟩, I, O, φ)
    φ̄ ← 0
    for e ∈ E do                                   ▷ these are edges
        k ← O[head(e)]
        for u in tail(e) do
            k ← k ⊗ I[u]
        end for
        φ̄ ← φ̄ + kφ(e)
    end for
    return φ̄                                       ▷ expected feature vector
end function
```

# Traversals

Viterbi derivation

1. start from the goal (root)
2. recursively rewrite every symbol $v$ by solving

$$e^\star = \arg\max_{e \in BS(v)} w(e) \bigotimes_{u \in \mathrm{tail}(e)} I(u)$$

# Traversals

Viterbi derivation

1. start from the goal (root)
2. recursively rewrite every symbol $v$ by solving

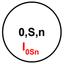
$$e^{\star} = \arg\max_{e \in BS(v)} w(e) \bigotimes_{u \in \text{tail}(e)} I(u)$$

Sampling

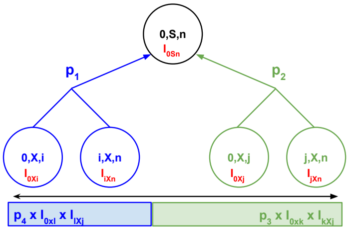
1. start from the goal (root)
2. recursively rewrite every symbol $v$ by solving

$$E \sim P(e|v) = \begin{cases} \bar{0} & \text{if } e \notin BS(v) \\ \frac{w(e) \bigotimes_{u \in \text{tail}(e)} I(u)}{I(v)} & \text{otherwise} \end{cases}$$

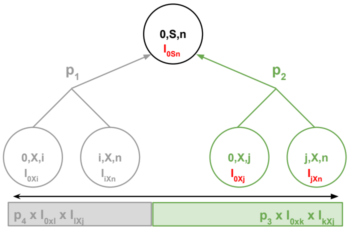
# Viterbi



# Viterbi

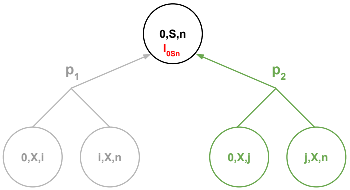

0,S,n
I0Sn
p1
p2
0,X,i
I0Xi
i,X,n
IiXn
0,X,j
I0Xj
j,X,n
IjXn
p4 x I0xl x IlXj
p3 x I0xk x IkXj

# Viterbi

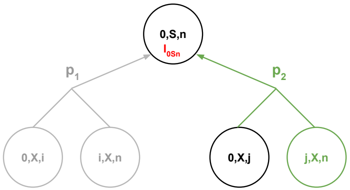


0,S,n
I0Sn
p1
p2
0,X,i
I0Xi
i,X,n
IiXn
0,X,j
I0Xj
j,X,n
IjXn
p4 x I0xl x IlXj
p3 x I0xk x IkXj

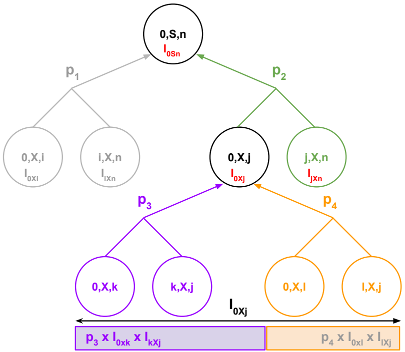
# Viterbi



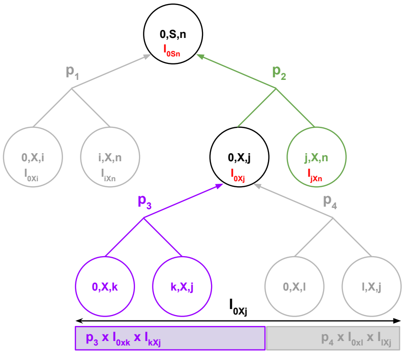
# Viterbi



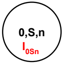
# Viterbi


0,S,n
l0Sn
p1
p2
0,X,i
l0Xi
i,X,n
liXn
0,X,j
l0Xj
j,X,n
ljXn
p3
p4
0,X,k
k,X,j
0,X,l
l,X,j
l0Xj
p3 x l0xk x lkXj
p4 x l0xl x llXj

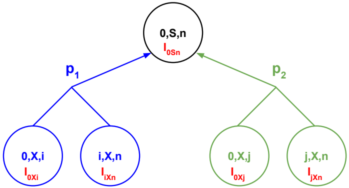
# Viterbi


0,S,n
l0Sn
p1
p2
0,X,i
l0Xi
i,X,n
liXn
0,X,j
l0Xj
j,X,n
ljXn
p3
p4
0,X,k
k,X,j
0,X,l
l,X,j
l0Xj
p3 x l0xk x lkXj
p4 x l0xl x llXj

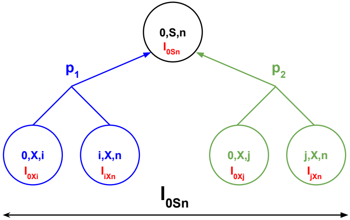
# Sampling



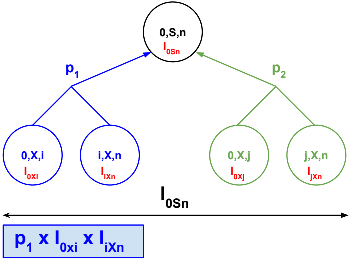
# Sampling



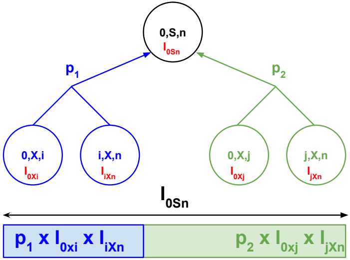
# Sampling



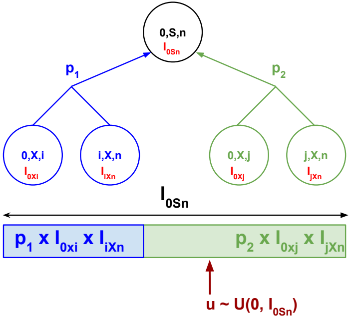
# Sampling


0,S,n
I0Sn
p1
p2
0,X,i
I0Xi
i,X,n
IiXn
0,X,j
I0Xj
j,X,n
IjXn
I0Sn
p1 x I0xi x IiXn

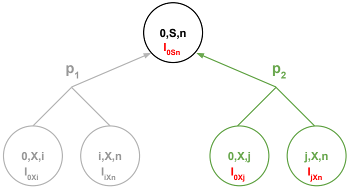
# Sampling


0,S,n
$I_{0Sn}$
$p_1$
$p_2$
0,X,i
$I_{0Xi}$
i,X,n
$I_{iXn}$
0,X,j
$I_{0Xj}$
j,X,n
$I_{jXn}$
$I_{0Sn}$
$p_1 \times I_{0xi} \times I_{iXn}$
$p_2 \times I_{0xj} \times I_{jXn}$

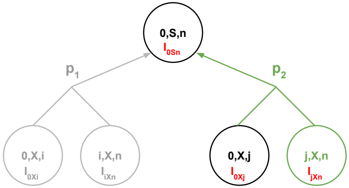
# Sampling


0,S,n
l0Sn
p1
p2
0,X,i
l0Xi
i,X,n
liXn
0,X,j
l0Xj
j,X,n
ljXn
l0Sn
p1 x l0xi x liXn
p2 x l0xj x ljXn
u ~ U(0, l0Sn)

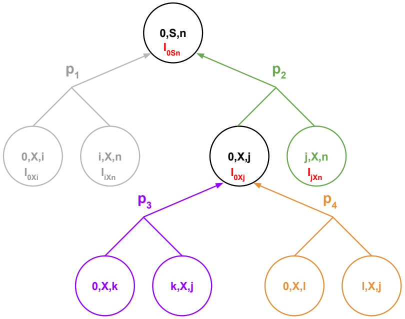
# Sampling



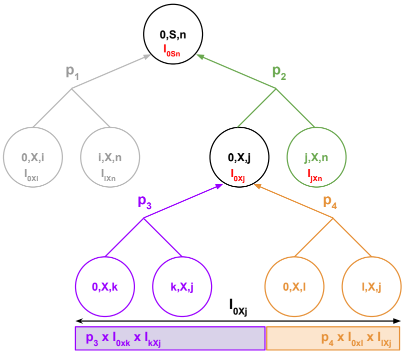
# Sampling



# Sampling

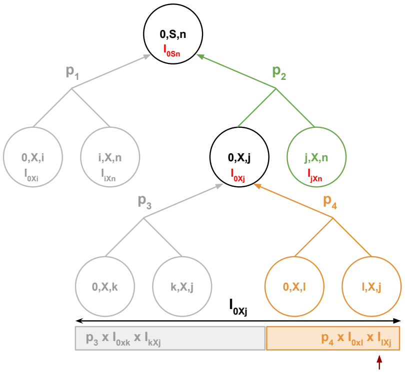

0,S,n
I0Sn
p1
p2
0,X,i
I0Xi
i,X,n
IiXn
0,X,j
I0Xj
j,X,n
IjXn
p3
p4
0,X,k
k,X,j
0,X,l
l,X,j

# Sampling



0,S,n
l0Sn
p1
p2
0,X,i
l0Xi
i,X,n
liXn
0,X,j
l0Xj
j,X,n
ljXn
p3
p4
0,X,k
k,X,j
0,X,l
l,X,j
l0Xj
p3 x l0xk x lkXj
p4 x l0xl x llXj

# Sampling


0,S,n
$I_{0Sn}$
$p_1$
$p_2$
0,X,i
$I_{0Xi}$
i,X,n
$I_{iXn}$
0,X,j
$I_{0Xj}$
j,X,n
$I_{jXn}$
$p_3$
$p_4$
0,X,k
k,X,j
0,X,l
l,X,j
$I_{0Xj}$
$p_3 \times I_{0xk} \times I_{kXj}$
$p_4 \times I_{0xl} \times I_{lXj}$

# Roadmap to project 2

1. understand the parser
   - check notes and notebook
2. implement the constrained forest $\mathcal{D}_n(x)$
3. implement feature functions
4. implement hypergraph algorithms
   - topsort, inside, outside, expected features
5. iterate over mini-batches of data making gradient updates

# Questions?

# References I

Phil Blunsom and Miles Osborne. Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1023`.

Phil Blunsom, Trevor Cohn, and Miles Osborne. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1024`.

# References II

Chris Dyer, Jonathan H. Clark, Alon Lavie, and Noah A. Smith. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 409–419, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1042`.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL `http://dx.doi.org/10.1162/089976602760128018`.

# References III

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`.

Gideon Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112, Rochester, New York, April 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N/N07/N07-2028`.

# References IV

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219884. URL http://www.aclweb.org/anthology/P05-1044.

Charles Sutton and Andrew McCallum. Piecewise training for undirected models. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 568–575, Arlington, Virginia, United States, 2005. AUAI Press. ISBN 0-9749039-1-4. URL http://dl.acm.org/citation.cfm?id=3020336.3020405.