

Latent-variable CRF for SMT

Wilker Aziz

Universiteit van Amsterdam

`w.aziz@uva.nl`

May 23, 2017

Conditional random field

Lafferty et al. [2001]

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

- ϕ 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

As₁ meninas**a**₂ for**am**₃ pra lá₄ \leftrightarrow The₁ girls₂ went₃ over₄ there₄

Apague₁ a₂ luz₃ \leftrightarrow Switch₁ the₂ light₃ off₁

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)

$$\mathcal{L}(w|x, y) = \log P(y|x, w) \quad (2)$$

$$= w^\top \phi(x, y) - \log Z(x|w) \quad (3)$$

Maximum likelihood estimation

Likelihood of an observation (x, y)

$$\mathcal{L}(w|x, y) = \log P(y|x, w) \quad (2)$$

$$= w^\top \phi(x, y) - \log Z(x|w) \quad (3)$$

Gradient-based optimisation

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

Maximum likelihood estimation

Likelihood of an observation (x, y)

$$\mathcal{L}(w|x, y) = \log P(y|x, w) \quad (2)$$

$$= w^\top \phi(x, y) - \log Z(x|w) \quad (3)$$

Gradient-based optimisation

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

Expected features should match the features of the observation

LV-CRF

Model

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

- 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)

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

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

Gradient-based optimisation [Mann and McCallum, 2007]

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

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

MLE for LV-CRF

Likelihood of an observation (x, y)

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

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

Gradient-based optimisation [Mann and McCallum, 2007]

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

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

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 *hieroglyph* grammar
- ϕ 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 $\text{yield}(d')$
- $r_{s,t}$ 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

$$S \rightarrow X$$

$$X \rightarrow X X$$

$$X \rightarrow x/y \text{ for all } x \in \Sigma \text{ and } y \in \Delta$$

$$X \rightarrow \epsilon/y \text{ for all } y \in \Delta$$

$$X \rightarrow x/\epsilon \text{ for all } x \in \Sigma$$

Global normalisation

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

Global normalisation

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

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

- $\mathcal{D}(x)$ is typically infinite

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

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 infinite

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

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)

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

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

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

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

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

What do we need?

An ITG parser in order to obtain

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

What do we need?

An ITG parser in order to obtain

- ➊ $\mathcal{D}(x)$
(potentially infinite) set of derivations over target strings that align to the source string x
- ➋ $\mathcal{D}_n(x)$
finite set of derivations over target strings that align to the source string x where the target string is no longer than n words
- ➌ $\mathcal{D}(x, y)$
set of derivations of the string pair (x, y)
- ➍ expected feature vector of observations (x, y)

What do we need?

An ITG parser in order to obtain

- ➊ $\mathcal{D}(x)$
(potentially infinite) set of derivations over target strings that align to the source string x
- ➋ $\mathcal{D}_n(x)$
finite set of derivations over target strings that align to the source string x where the target string is no longer than n words
- ➌ $\mathcal{D}(x, y)$
set of derivations of the string pair (x, y)
- ➍ expected feature vector of observations (x, y)
- ➎ expected feature vector of an observation x

Feature functions

Featurise edges in the support of the CRF

- support: the finite set $D_n(x)$
- edges in $D(x, y)$ correspond to a subset of edges in $D_n(x)$ annotated with some extra span information necessary in order to restrict the set to trees that generate y

Feature types

- lexical: e.g. translation pairs;
- segmentation: e.g. number of deletions, number of translation rules, number of insertions, number of binary rules;
- word order: e.g. number of inversions, skip-bigrams.

Notation

Using a hypergraph view

- u, v, s are nodes
- e is an edge
- $\text{head}(e)$ is a node
- $\text{tail}(e)$ is a sequence of nodes
- if $u \in \text{tail}(e)$ and $v \in \text{tail}(e)$, 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

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

For acyclic hypergraphs

Outside

The OUTSIDE recursion

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

For acyclic hypergraphs

Topsort

```
1: function TOPSORT( $G = \langle V, \langle E, w \rangle \rangle$ )
2:    $S = \{v \in V : BS(v) = \emptyset\}$   $\triangleright$  nodes with no dependencies
3:    $D = \{v \mapsto \{u : \exists e \in BS(v) \wedge u \in \text{tail}(e)\} : v \in V\}$ 
4:    $\triangleright$  a node depends on all of its children
5:    $L = \langle \rangle$   $\triangleright$  top-sorted nodes
6:   while  $S \neq \emptyset$  do
7:      $u \leftarrow \text{pop}(S)$   $\triangleright$  remove and return a node from  $S$ 
8:      $L \leftarrow L + \langle u \rangle$   $\triangleright$  append  $u$  to  $L$ 
9:     for  $e$  in  $FS(u)$  do  $\triangleright$  outgoing edges from  $u$ 
10:       $v \leftarrow \text{head}(e)$   $\triangleright$  parent of  $u$  in  $e$ 
11:       $D(v) \leftarrow D(v) \setminus \{u\}$   $\triangleright$  remove  $u$  from  $D(v)$ 
12:      if  $D(v) == \emptyset$  then  $\triangleright v$ 's dependencies have been sorted
13:         $S \leftarrow S \cup \{v\}$ 
14:      end if
15:    end for
16:  end while
17:  return  $L$ 
18: end function
```

Inside

```
1: function INSIDE( $G = \langle V, \langle E, w \rangle \rangle$ )
2:   for  $v$  in TOPSORT( $G$ ) do                                ▷ visit nodes bottom-up
3:     if  $BS(v) == \emptyset$  then
4:        $I[v] \leftarrow \bar{1}$                                     ▷ leaves
5:     else
6:        $I[v] \leftarrow \bar{0}$ 
7:       for  $e \in BS(v)$  do
8:          $k \leftarrow w(e)$                                 ▷ include the edge's own weight
9:         for  $u$  in tail( $e$ ) do
10:           $k \leftarrow k \otimes I[u]$ 
11:        end for
12:         $I[v] \leftarrow I[v] \oplus k$                         ▷ accumulate for each edge
13:      end for
14:    end if
15:  end for
16:  return  $I$ 
17: end function
```

Outside

```
1: function OUTSIDE( $G = \langle V, \langle E, w \rangle \rangle, I, \text{root}$ )
2:    $O[v] \leftarrow \bar{0}$  for  $v \in V$ 
3:    $O[\text{root}] \leftarrow \bar{1}$ 
4:   for  $v$  in REVERSE(TOPSORT( $G$ )) do
5:     for  $e \in BS(v)$  do
6:       for  $u \in \text{tail}(e)$  do
7:          $k \leftarrow w(e) \otimes O[v]$ 
8:         for  $s$  in tail( $e$ ) do
9:           if  $u \neq s$  then
10:             $k \leftarrow k \otimes I[s]$ 
11:          end if
12:        end for
13:         $O[u] \leftarrow O[u] \oplus k$ 
14:      end for
15:    end for
16:  end for
17:  return  $O$ 
18: end function
```

▷ this is the goal node

▷ visit nodes top-down

▷ q 's incoming edges

▷ children of v in e

▷ siblings of u in e

▷ u itself is excluded

▷ accumulate it for u

Expected features

Expectation semiring [Li and Eisner, 2009]

```
1: function EXPECTEDFEATURES( $G = \langle V, \langle E, w \rangle \rangle, I, O, \phi$ )
2:    $\bar{\phi} \leftarrow 0$ 
3:   for  $e \in E$  do ▷ these are edges
4:      $k \leftarrow O[\text{head}(e)]$ 
5:     for  $u$  in  $\text{tail}(e)$  do
6:        $k \leftarrow k \otimes I[u]$ 
7:     end for
8:      $\bar{\phi} \leftarrow \bar{\phi} + k\phi(e)$ 
9:   end for
10:  return  $\bar{\phi}$  ▷ proportional to expected feature vector
11: end function
```

Expectations manipulate quantities in a Probability semiring

- where weights are normalised probabilities
- \otimes is multiplication
- \oplus is sum

Traversals

Viterbi derivation

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

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

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

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

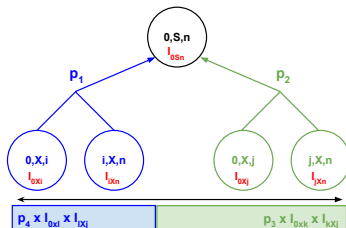
Viterbi

Use Inside computed in the Max-Times semiring!



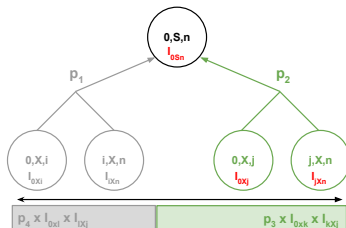
Viterbi

Use Inside computed in the Max-Times semiring!



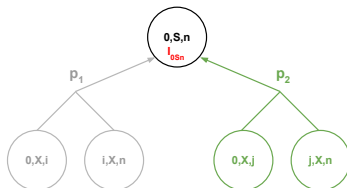
Viterbi

Use Inside computed in the Max-Times semiring!



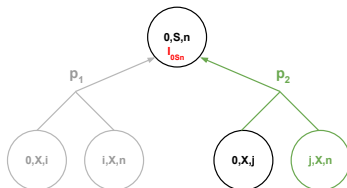
Viterbi

Use Inside computed in the Max-Times semiring!



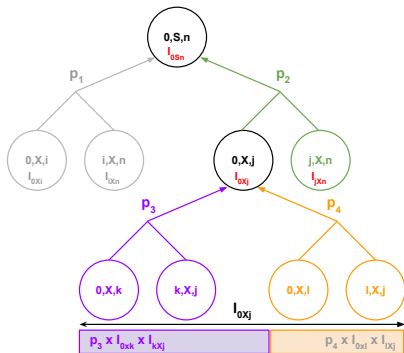
Viterbi

Use Inside computed in the Max-Times semiring!



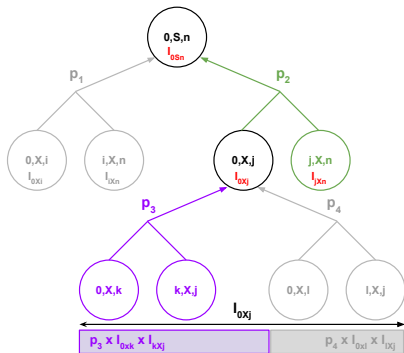
Viterbi

Use Inside computed in the Max-Times semiring!



Viterbi

Use Inside computed in the Max-Times semiring!



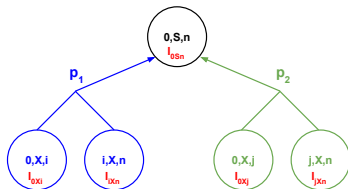
Sampling

Use Inside computed in the Sum-Times semiring!



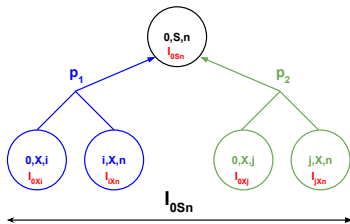
Sampling

Use Inside computed in the Sum-Times semiring!



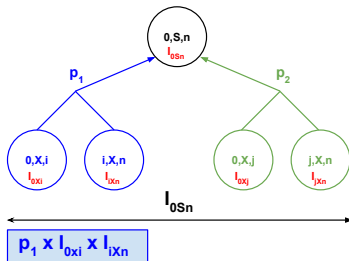
Sampling

Use Inside computed in the Sum-Times semiring!



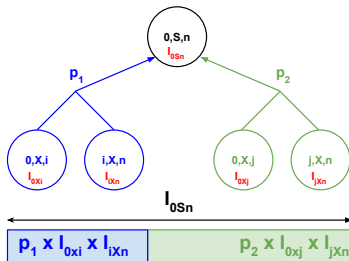
Sampling

Use Inside computed in the Sum-Times semiring!



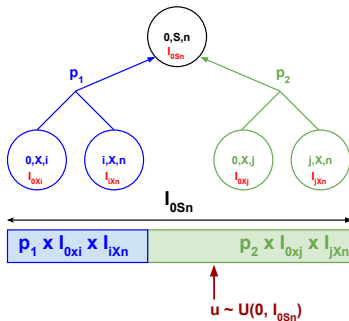
Sampling

Use Inside computed in the Sum-Times semiring!



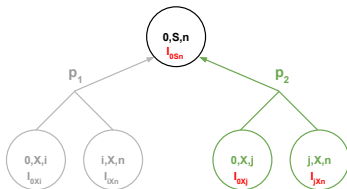
Sampling

Use Inside computed in the Sum-Times semiring!



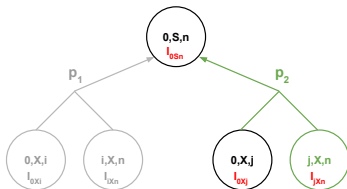
Sampling

Use Inside computed in the Sum-Times semiring!



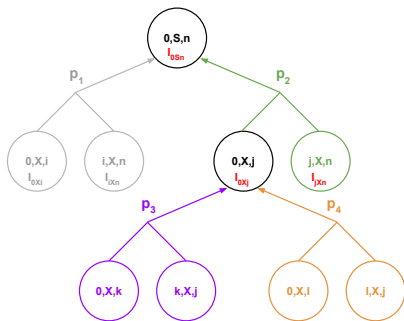
Sampling

Use Inside computed in the Sum-Times semiring!



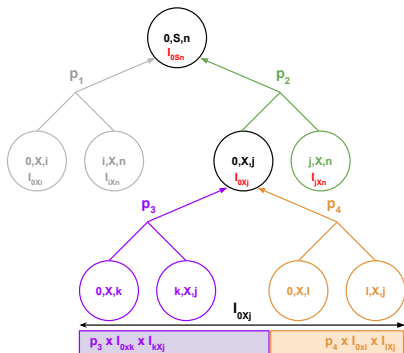
Sampling

Use Inside computed in the Sum-Times semiring!



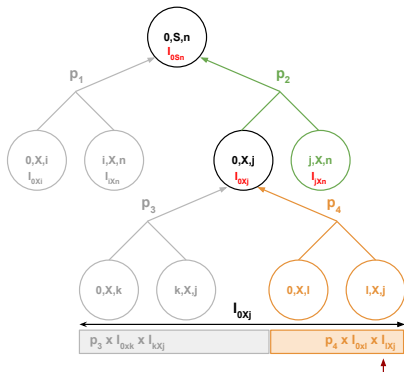
Sampling

Use Inside computed in the Sum-Times semiring!



Sampling

Use Inside computed in the Sum-Times semiring!



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>.
- Zhifei Li and Jason Eisner. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Singapore, August 2009. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/D/D09/D09-1005>.

References IV

- 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>.
- 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>.

References V

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>.