Neural Language Models with Latent Syntax

Thesis Master of Logic

Daan van Stigt

May 24, 2019

Institute for Logic, Language and Computation (ILLC)

Neural Language Models with Latent Syntax

Neural Language Models with Latent Syntax

$$p(x)$$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p\ (x, y)$$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p\,(x, y)$$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p\,(x, y)$$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p_{\boldsymbol{\theta}}(x, y)$$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p_{\theta}(x, y)$$

$\vdots$

The hungry cat meows.

(NP The hungry cat meows .)
(S (NP The (ADJP hungry) cat) (VP meows) .)
(S (NP The hungry cat meows) .)
(S (NP The hungry cat) (VP meows) .)
(SINV (FRAG The hungry cat) (NP meows) .)

$\vdots$

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p_{\boldsymbol{\theta}}(x, y)$$

	⋮
	(NP The hungry cat meows .)
	(S (NP The (ADJP hungry) cat) (VP meows) .)
The hungry cat meows.	(S (NP The hungry cat meows) .)
	(S (NP The hungry cat) (VP meows) .)
	(SINV (FRAG The hungry cat) (NP meows) .)
	⋮

Neural Language Models with Latent Syntax

$$p(x) = \sum_{y} p_{\theta}(x, y)$$

⋮

(NP The hungry cat meows .)

(S (NP The (ADJP hungry) cat) (VP meows) .)

The hungry cat meows. (S (NP The hungry cat meows) .)

(S (NP The hungry cat) (VP meows) .)

(SINV (FRAG The hungry cat) (NP meows) .)

⋮

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \dots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

the hungry cat meows *hungry meows cat the*

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

$$p(\textit{the hungry cat meows}) > p(\textit{hungry meows cat the})$$

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

from some vocabulary $V = \{\textit{cat}, \textit{dog}, a, \ldots, \textit{The}\}$.

- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

$$p(\textit{the hungry cat meows}) > p(\textit{hungry meows cat the})$$

Both are sequences of words; only one is a sentence.

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

$$p(\textit{the hungry cat meows}) > p(\textit{hungry meows cat the})$$

  Both are sequences of words; only one is a sentence.
- More subtle

$$\textit{The hungry cat meows} \qquad \textit{The hungry cat meow}$$

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

$$p(\textit{the hungry cat meows}) > p(\textit{hungry meows cat the})$$

  Both are sequences of words; only one is a sentence.
- More subtle

$$p(\textit{The hungry cat meows}) > p(\textit{The hungry cat meow})$$

Probability of a sentence

- From statistical viewpoint a sentence is a sequence of words

$$\textit{The hungry cat meows} \cong \langle \textit{The}, \textit{hungry}, \textit{cat}, \textit{meows} \rangle$$

  from some vocabulary $V = \{cat, dog, a, \ldots, The\}$.
- In isolation this question looks odd:

$$p(\textit{The hungry cat meows .}) = ?$$

- Comparatively it makes more sense:

$$p(\textit{the hungry cat meows}) > p(\textit{hungry meows cat the})$$

  Both are sequences of words; only one is a sentence.
- More subtle

$$p(\textit{The hungry cat meows}) > p(\textit{The hungry cat meow})$$

  Knowing the syntactic structure of the sentence could help.

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

The hungry cat apparently meows.

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

The hungry cat apparently meows.

**The hungry apparently cat meows.*

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

The hungry cat apparently meows.
(S (NP *The hungry cat*) *apparently* (VP *meows*) .)

* *The hungry apparently cat meows.*

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

The hungry cat apparently meows.
(S (NP *The hungry cat*) *apparently* (VP *meows*) .)

**The hungry apparently cat meows.*
(S (NP *The hungry apparently cat*) (VP *meows*) .)

Syntactic structure

Consider inserting the word apparently in the sentence

The hungry cat meows.

The hungry cat apparently meows.

(S (NP *The hungry cat*) *apparently* (VP *meows*) .)

**The hungry apparently cat meows.*

(S (NP *The hungry apparently cat*) (VP *meows*) .)

$$p(\textit{The hungry cat apparently meows.}) > p(\textit{The hungry apparently cat meows.})$$

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

- We are printerested in knowing only $p(x)$

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

- We are printerested in knowing only $p(x)$
- Need to compute

$$p(x) = \sum_{y \in \mathcal{Y}(x)} p_\theta(x, y)$$

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

- We are printerested in knowing only $p(x)$
- Need to compute

$$p(x) = \sum_{y \in \mathcal{Y}(x)} p_\theta(x, y)$$

- The large sum is in the way.

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

- We are printerested in knowing only $p(x)$
- Need to compute

$$p(x) = \sum_{y \in \mathcal{Y}(x)} p_\theta(x, y)$$

- The large sum is in the way.
- Idea: use only some of the stuctures y to approximate the sum.

Syntactic language model

Given a model that assigns probabilities $p_\theta(x, y)$ to both x and y

- We are printerested in knowing only $p(x)$
- Need to compute

$$p(x) = \sum_{y \in \mathcal{Y}(x)} p_\theta(x, y)$$

- The large sum is in the way.
- Idea: use only some of the stuctures y to approximate the sum.
- Which stuctures? Those proposed by a (separate) conditional model $q(y \mid x)$.

Approximate marginal

Solution: rewrite the sum as an expectation under q:

$$p(x) = \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \frac{p_\theta(x, y)}{q(y \mid x)}$$

Approximate marginal

Solution: rewrite the sum as an expectation under q:

$$\begin{aligned} p(x) &= \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \frac{p_\theta(x, y)}{q(y \mid x)} \\ &= \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right] \end{aligned}$$

Approximate marginal

Solution: rewrite the sum as an expectation under q:

$$p(x) = \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \frac{p_\theta(x, y)}{q(y \mid x)}$$
$$= \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

- Estimate this expectation with samples from q

Approximate marginal

Solution: rewrite the sum as an expectation under q:

$$
\begin{aligned}
p(x) &= \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \frac{p_\theta(x, y)}{q(y \mid x)} \\
&= \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]
\end{aligned}
$$

- Estimate this expectation with samples from q
- Sample trees $y^{(1)}, \ldots, y^{(K)}$ from $q(y \mid x)$ and estimate

$$
\mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right] \approx \sum_{k=1}^{K} \frac{p_\theta(x, y^{(k)})}{q(y^{(k)} \mid x)}
$$

Approximate marginal

Solution: rewrite the sum as an expectation under q:

$$p(x) = \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \frac{p_\theta(x, y)}{q(y \mid x)}$$
$$= \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

- Estimate this expectation with samples from q
- Sample trees $y^{(1)}, \ldots, y^{(K)}$ from $q(y \mid x)$ and estimate

$$\mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right] \approx \sum_{k=1}^{K} \frac{p_\theta(x, y^{(k)})}{q(y^{(k)} \mid x)}$$

- A much smaller sum!

Approximate posterior

We call the model $q(y \mid x)$ an approximate posterior

Approximate posterior

We call the model $q(y \mid x)$ an approximate posterior

- Approximates the true posterior, which is unavaillable. Recall

$$p_\theta(y \mid x) = \frac{p_\theta(x, y)}{p(x)}$$

Approximate posterior

We call the model $q(y \mid x)$ an approximate posterior

- Approximates the true posterior, which is unavaillable. Recall

$$p_\theta(y \mid x) = \frac{p_\theta(x, y)}{p(x)}$$

- The closer $q(y \mid x)$ is to $p_\theta(y \mid x)$ the better the approximation

Approximate posterior

We call the model $q(y \mid x)$ an approximate posterior

- Approximates the true posterior, which is unavaillable. Recall

$$p_\theta(y \mid x) = \frac{p_\theta(x, y)}{p(x)}$$

- The closer $q(y \mid x)$ is to $p_\theta(y \mid x)$ the better the approximation
- A conditional distribution over trees...

Approximate posterior

We call the model $q(y \mid x)$ an approximate posterior

- Approximates the true posterior, which is unavaillable. Recall

$$p_\theta(y \mid x) = \frac{p_\theta(x, y)}{p(x)}$$

- The closer $q(y \mid x)$ is to $p_\theta(y \mid x)$ the better the approximation
- A conditional distribution over trees... A probabilistic parser!

Key ideas of thesis

We let $p_\theta(x, y)$ be an RNNG [Dyer et al., 2016]

- competitive language model sensitive to syntactic phenomena
- sum intractable (approximation possible)
- training limited to annotated data

Key ideas of thesis

We let $p_\theta(x, y)$ be an RNNG [Dyer et al., 2016]

- competitive language model sensitive to syntactic phenomena
- sum intractable (approximation possible)
- training limited to annotated data

Goal: approximate learning of both with $q(y \mid x)$

- allows estimation from data without annotation
- induce structure y without any examples

Key ideas of thesis

We let $p_\theta(x, y)$ be an RNNG [Dyer et al., 2016]

- competitive language model sensitive to syntactic phenomena
- sum intractable (approximation possible)
- training limited to annotated data

Contribution: let $q(y \mid x)$ be a CRF parser

- Strong independence assumptions
- Efficient computation of key quantities

Goal: approximate learning of both with $q(y \mid x)$

- allows estimation from data without annotation
- induce structure y without any examples

$$p(x) = \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

$$p(x) = \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a)$$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \dots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

The

$p(x_1)$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

The	hungry
$p(x_1)$	$p(x_2 \mid x_1)$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

The	hungry	cat
$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

The	hungry	cat	meows
$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP The

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \dots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP The hungry

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP The hungry cat

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP The hungry cat)

Recurrent Neural Network Grammar

Models the sequence of actions $a = \langle a_1, \ldots, a_T \rangle$ that build sentence x together with tree y

$$p_\theta(x, y) = p_\theta(a) = \prod_{t=1}^{T} p_\theta(a_t \mid a_{<t})$$

Conditions on entire history.

Recall: generating only x:

	The	hungry	cat	meows	.
$p(x) =$	$p(x_1)$	$p(x_2 \mid x_1)$	$p(x_3 \mid x_{<3})$	$p(x_4 \mid x_{<4})$	$p(x_5 \mid x_{<5})$

Generate both x and y:

(S (NP The hungry cat) (VP meows) .)

Transition system

Stack	Action

Transition system

Stack	Action
	OPEN(S)
(S	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	

Transition system

Stack			Action
			OPEN(S)
(S			OPEN(NP)
(S	(NP		GEN(*The*)
(S	(NP	*The*	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	

Transition system

Stack					Action
					OPEN(S)
(S					OPEN(NP)
(S	(NP				GEN(*The*)
(S	(NP	*The*			GEN(*hungry*)
(S	(NP	*The*	*hungry*		GEN(*cat*)
(S	(NP	*The*	*hungry*	*cat*	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	GEN(*cat*)
(S (NP *The* *hungry* *cat*	REDUCE
(S (NP *The hungry cat*)	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	GEN(*cat*)
(S (NP *The* *hungry* *cat*	REDUCE
(S (NP *The hungry cat*)	OPEN(VP)
(S (NP *The hungry cat*) (VP	

Transition system

Stack						Action
						OPEN(S)
(S						OPEN(NP)
(S	(NP					GEN(*The*)
(S	(NP	*The*				GEN(*hungry*)
(S	(NP	*The*	*hungry*			GEN(*cat*)
(S	(NP	*The*	*hungry*	*cat*		REDUCE
(S	(NP *The hungry cat*)					OPEN(VP)
(S	(NP *The hungry cat*)	(VP				GEN(*meows*)
(S	(NP *The hungry cat*)	(VP	*meows*			

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	GEN(*cat*)
(S (NP *The* *hungry* *cat*	REDUCE
(S (NP *The hungry cat*)	OPEN(VP)
(S (NP *The hungry cat*) (VP	GEN(*meows*)
(S (NP *The hungry cat*) (VP *meows*	REDUCE
(S (NP *The hungry cat*) (VP *meows*)	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	GEN(*cat*)
(S (NP *The* *hungry* *cat*	REDUCE
(S (NP *The hungry cat*)	OPEN(VP)
(S (NP *The hungry cat*) (VP	GEN(*meows*)
(S (NP *The hungry cat*) (VP *meows*	REDUCE
(S (NP *The hungry cat*) (VP *meows*)	GEN(.)
(S (NP *The hungry cat*) (VP *meows*) .	

Transition system

Stack	Action
	OPEN(S)
(S	OPEN(NP)
(S (NP	GEN(*The*)
(S (NP *The*	GEN(*hungry*)
(S (NP *The* *hungry*	GEN(*cat*)
(S (NP *The* *hungry* *cat*	REDUCE
(S (NP *The hungry cat*)	OPEN(VP)
(S (NP *The hungry cat*) (VP	GEN(*meows*)
(S (NP *The hungry cat*) (VP *meows*	REDUCE
(S (NP *The hungry cat*) (VP *meows*)	GEN(.)
(S (NP *The hungry cat*) (VP *meows*) .	REDUCE
(S (NP *The hungry cat*) (VP *meows*) .)	

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

A composition function computes a single representation

(NP *The hungry cat*) from (NP *The* *hungry* *cat*

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

A composition function computes a single representation

(NP *The hungry cat*) from (NP *The* *hungry* *cat*

The RNN updates with the composed representation

(S

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

A composition function computes a single representation

(NP *The hungry cat*) from (NP *The* *hungry* *cat*

The RNN updates with the composed representation

(S (NP *The hungry cat*)

Parametrization

An RNN encodes the stack, and backtracks upon reduce action

(S (NP *The* *hungry* *cat*

A composition function computes a single representation

(NP *The hungry cat*) from (NP *The* *hungry* *cat*

The RNN updates with the composed representation

(S (NP *The hungry cat*)

The RNNG is an RNN that recursively compresses its history.

$$p(x) = \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

$$p(x) = \mathbb{E}_q \left[\frac{p_\theta(x, y)}{q(y \mid x)} \right]$$

CRF parser

A globally normalized model

$$q(y \mid x) = \frac{\Psi(x, y)}{Z(x)},$$

where

- $\Psi(x, y) \geq 0$ scores trees
- $Z(x) = \sum_{y \in \mathcal{Y}(x)} \Psi(x, y)$ is a global normalizer

CRF parser

A globally normalized model

$$q(y \mid x) = \frac{\Psi(x, y)}{Z(x)},$$

where

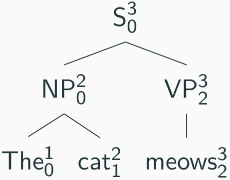
- $\Psi(x, y) \geq 0$ scores trees
- $Z(x) = \sum_{y \in \mathcal{Y}(x)} \Psi(x, y)$ is a global normalizer

We factorize Ψ over the parts of the tree y_c:

$$\Psi(x, y) = \prod_{c=1}^{C} \psi(x, y_c),$$

- Parts $y_c = (A, i, j)$ are labeled spans, *e.g.* (NP, 0, 2)
- Factorization allows efficient computation of $\sum$!

Factorization



Scoring entire tree $\Psi(x, y)$

(S, 0, 3)
(NP, 0, 2)
(VP, 2, 3)

Scoring parts of tree $\psi(x, y_c)$

Parametrization

- Compute score $\psi(x, y_c)$ with a neural network following Stern et al. [2017]
- Use feedfoward neural network to compute the span score

$$\log \psi(x, y_c) = [\textsc{ffn}(\mathbf{s}_{ij})]_A.$$

- Use bidirectional RNN to compute $\mathbf{s}_{ij}$

$$\mathbf{s}_{ij} = [\mathbf{f}_j - \mathbf{f}_i; \mathbf{b}_i - \mathbf{b}_j],$$

  where $\mathbf{f}$ and $\mathbf{b}$ forward and backward vectors.
- Very minimalistic setup!

Parametrization

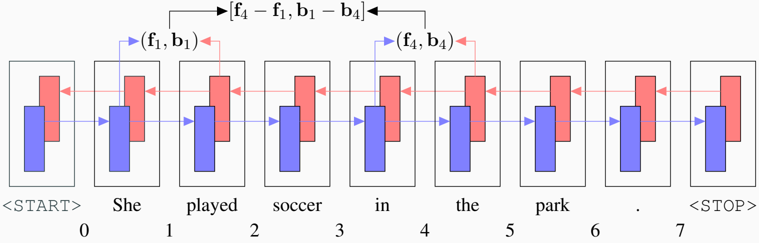


Figure 1: Span representation $\mathbf{s}_{ij}$ from $\mathbf{f}$ and $\mathbf{b}$. Figure by Gaddy et al. [2018].

Inference

The factorization over spans model allows efficient solutions to four related problems:

- Compute normalizer $Z(x) = \sum_y \Psi(x, y)$
- Obtain best parse $\hat{y} = \arg\max_y q(y \mid x)$
- Sample a tree $y \sim q(y \mid x)$
- Compute the entropy $\mathrm{H}(q(y \mid x))$

Inference

The factorization over spans model allows efficient solutions to four related problems:

- Compute normalizer $Z(x) = \sum_y \Psi(x, y)$
- Obtain best parse $\hat{y} = \arg\max_y q(y \mid x)$
- Sample a tree $y \sim q(y \mid x)$
- Compute the entropy $\mathrm{H}(q(y \mid x))$

These problems involve either search or summation over all trees!

Inference

The factorization over spans model allows efficient solutions to four related problems:

- Compute normalizer $Z(x) = \sum_y \Psi(x, y)$
- Obtain best parse $\hat{y} = \arg\max_y q(y \mid x)$
- Sample a tree $y \sim q(y \mid x)$
- Compute the entropy $\mathrm{H}(q(y \mid x))$

These problems involve either search or summation over all trees!

- Can be solved efficiently by the inside-outside algorithm.

Entropy

$$H(q(y \mid x)) := -\sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log q(y \mid x)$$

Entropy

$$
\begin{aligned}
H(q(y \mid x)) &:= -\sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log q(y \mid x) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v)
\end{aligned}
$$

Entropy

$$
\begin{aligned}
H(q(y \mid x)) &:= -\sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log q(y \mid x) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in \mathcal{V}(x)} \mathbf{1}(v \in y) \log \psi(x, v)
\end{aligned}
$$

Entropy

$$
\begin{aligned}
H(q(y \mid x)) &:= -\sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log q(y \mid x) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in \mathcal{V}(x)} \mathbf{1}(v \in y) \log \psi(x, v) \\
&= \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \sum_{y \in \mathcal{Y}(x)} \mathbf{1}(v \in y) q(y \mid x)
\end{aligned}
$$

Entropy

$$
\begin{aligned}
H(q(y \mid x)) &:= -\sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log q(y \mid x) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in \mathcal{V}(x)} \mathbf{1}(v \in y) \log \psi(x, v) \\
&= \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \sum_{y \in \mathcal{Y}(x)} \mathbf{1}(v \in y) q(y \mid x) \\
&= \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \mu(v).
\end{aligned}
$$

Entropy

$$H(q(y \mid x)) = \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \mu(v)$$

Entropy

$$H(q(y \mid x)) = \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \mu(v)$$

The marginal of a node $v = (A, i, j)$

$$\mu(v) := \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \, \mathbf{1}(v \in y) = \mathbb{E}[\mathbf{1}(v \in Y)]$$

is the probability that v appears in y, according to $P(Y \mid X = x)$.

Entropy

$$H(q(y \mid x)) = \log Z(x) - \sum_{v \in \mathcal{V}(x)} \log \psi(x, v) \mu(v)$$

The marginal of a node $v = (A, i, j)$

$$\begin{aligned} \mu(v) &:= \sum_{y \in \mathcal{Y}(x)} q(y \mid x)\, \mathbf{1}(v \in y) = \mathbb{E}[\mathbf{1}(v \in Y)] \\ &= \frac{\alpha(A, i, j)\, \beta(A, i, j)}{Z(x)} \end{aligned}$$

is the probability that v appears in y, according to $P(Y \mid X = x)$.

Entropy

The entropy has a nice interpretation:

$$\begin{aligned} H(q(y \mid x)) &= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\ &= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log \Psi(x, y) \\ &= \log Z(x) - \mathbb{E}_p \left[\log \Psi(x, Y) \right] \end{aligned}$$

Entropy

The entropy has a nice interpretation:

$$
\begin{aligned}
H(q(y \mid x)) &= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log \Psi(x, y) \\
&= \log Z(x) - \mathbb{E}_p \left[\log \Psi(x, Y) \right]
\end{aligned}
$$

log-weight of all trees

Entropy

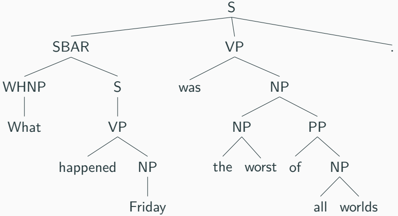
The entropy has a nice interpretation:

$$
\begin{aligned}
H(q(y \mid x)) &= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \sum_{v \in y} \log \psi(x, v) \\
&= \log Z(x) - \sum_{y \in \mathcal{Y}(x)} q(y \mid x) \log \Psi(x, y) \\
&= \log Z(x) - \mathbb{E}_p \left[\log \Psi(x, Y) \right]
\end{aligned}
$$

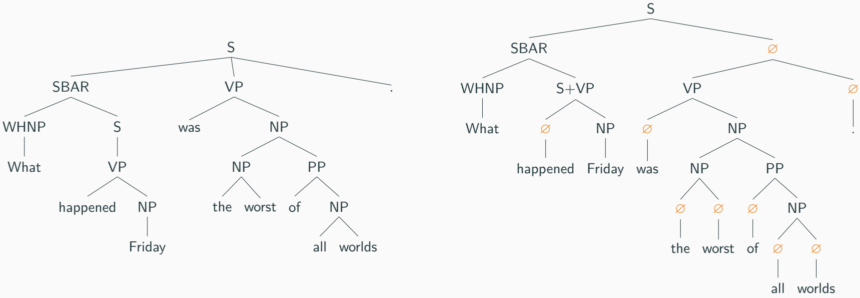
log-weight of all trees

log-weight of the expected tree

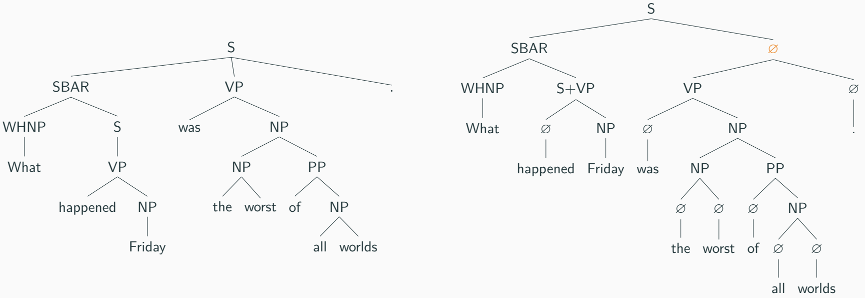
Dealing with *n*-ary trees



Dealing with *n*-ary trees



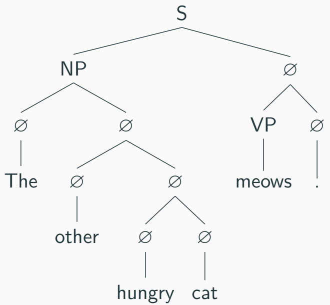
Dealing with *n*-ary trees

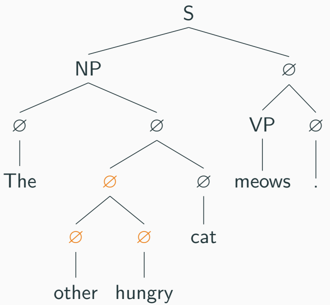


Derivational ambiguity

We treat the dummy label ∅ as any other label

- Absorbing the empty labels gives back an *n*-ary tree
- Causes derivational ambiguity!





(S (NP The other hungry cat) (VP meows) .).

$$p(x) = \mathbb{E}_{q_\lambda}\left[\frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right]$$

$$\log p(x) = \log \mathbb{E}_{q_\lambda}\left[\frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right]$$

$$\mathcal{L}(\theta, \lambda) = \log p(x) = \log \mathbb{E}_{q_\lambda}\left[\frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right]$$

$$\mathcal{L}(\theta, \lambda) = \log p(x) = \log \mathbb{E}_{q_\lambda}\left[\frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right]$$

$$\begin{aligned}\mathcal{L}(\theta, \lambda) = \log p(x) = \log \mathbb{E}_{q_\lambda}\left[\frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right] \\ \geq \mathbb{E}_{q_\lambda}\left[\log \frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right]\end{aligned}$$

Unsupervised learning

The objective:

$$\begin{aligned}
\mathcal{L}(\theta, \lambda) &\geq \mathbb{E}_{q_\lambda}\left[\log \frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right] \\
&= \mathbb{E}_{q_\lambda}[\log p_\theta(x, y)] - \mathbb{E}_{q_\lambda}[\log q_\lambda(y \mid x)] \\
&= \underbrace{\mathbb{E}_{q_\lambda}[\log p_\theta(x, y)]}_{\text{estimate with samples}} - \underbrace{\mathrm{H}(q_\lambda(y \mid x))}_{\text{exact computation with CRF!}}
\end{aligned}$$

Can we estimate θ and λ this way?

Unsupervised learning

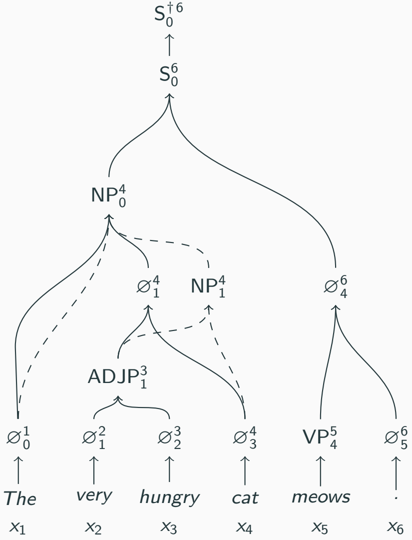
The objective:

$$
\begin{aligned}
\mathcal{L}(\theta, \lambda) &\geq \mathbb{E}_{q_\lambda}\left[\log \frac{p_\theta(x, y)}{q_\lambda(y \mid x)}\right] \\
&= \mathbb{E}_{q_\lambda}[\log p_\theta(x, y)] - \mathbb{E}_{q_\lambda}[\log q_\lambda(y \mid x)] \\
&= \underbrace{\mathbb{E}_{q_\lambda}[\log p_\theta(x, y)]}_{\text{estimate with samples}} - \underbrace{\mathrm{H}(q_\lambda(y \mid x))}_{\text{exact computation with CRF!}}
\end{aligned}
$$

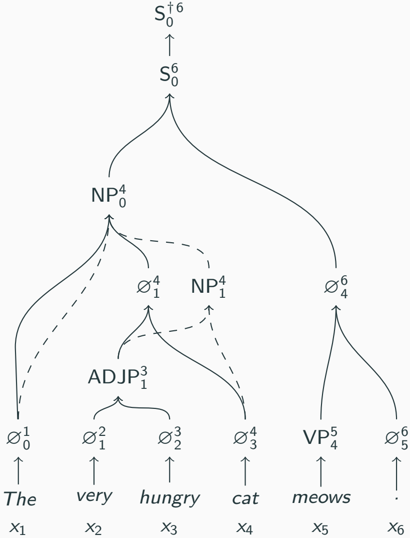
Can we estimate θ and λ this way? Almost there...

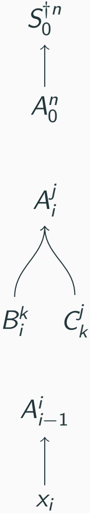
Backup slides

Compact parse forest



Compact parse forest





Inside algorithm

$$\alpha(A, i, j) = \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u)$$

Inside algorithm

sum over all incoming edges

$$\alpha(A,i,j) = \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u)$$

Inside algorithm

sum over all incoming edges

$$\alpha(A, i, j) = \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u)$$

weight of edge

Inside algorithm

sum over all incoming edges

$$\alpha(A, i, j) = \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u)$$

weight of edge

product of inside scores of the nodes in the tail of the edge

Inside algorithm

$$\alpha(A, i, j) = \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u)$$

$$= \bigoplus_{B \in \Lambda} \bigoplus_{C \in \Lambda} \bigoplus_{k=i+1}^{j-1} \omega(A, i, j) \otimes \alpha(B, i, k) \otimes \alpha(C, k, j)$$

Inside algorithm

$$
\begin{aligned}
\alpha(A,i,j) &= \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u) \\
&= \bigoplus_{B \in \Lambda} \bigoplus_{C \in \Lambda} \bigoplus_{k=i+1}^{j-1} \omega(A,i,j) \otimes \alpha(B,i,k) \otimes \alpha(C,k,j)
\end{aligned}
$$

Inside algorithm

$$
\begin{aligned}
\alpha(A, i, j) &= \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u) \\
&= \bigoplus_{B \in \Lambda} \bigoplus_{C \in \Lambda} \bigoplus_{k=i+1}^{j-1} \omega(A, i, j) \otimes \alpha(B, i, k) \otimes \alpha(C, k, j) \\
&= \omega(A, i, j) \otimes \bigoplus_{k=i+1}^{j-1} \bigoplus_{B \in \Lambda} \alpha(B, i, k) \otimes \bigoplus_{C \in \Lambda} \alpha(C, k, j)
\end{aligned}
$$

Inside algorithm

$$
\begin{aligned}
\alpha(A, i, j) &= \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u) \\
&= \bigoplus_{B \in \Lambda} \bigoplus_{C \in \Lambda} \bigoplus_{k=i+1}^{j-1} \omega(A, i, j) \otimes \alpha(B, i, k) \otimes \alpha(C, k, j) \\
&= \omega(A, i, j) \otimes \bigoplus_{k=i+1}^{j-1} \bigoplus_{B \in \Lambda} \alpha(B, i, k) \otimes \bigoplus_{C \in \Lambda} \alpha(C, k, j)
\end{aligned}
$$

Inside algorithm

$$
\begin{aligned}
\alpha(A, i, j) &= \bigoplus_{e \in I(v)} \omega(e) \otimes \bigotimes_{u \in T(e)} \alpha(u) \\
&= \bigoplus_{B \in \Lambda} \bigoplus_{C \in \Lambda} \bigoplus_{k=i+1}^{j-1} \omega(A, i, j) \otimes \alpha(B, i, k) \otimes \alpha(C, k, j) \\
&= \omega(A, i, j) \otimes \bigoplus_{k=i+1}^{j-1} \bigoplus_{B \in \Lambda} \alpha(B, i, k) \otimes \bigoplus_{C \in \Lambda} \alpha(C, k, j) \\
&= \omega(A, i, j) \otimes \bigoplus_{k=i+1}^{j-1} \sigma(i, k) \otimes \sigma(k, j)
\end{aligned}
$$

Outside algorithm

$$\beta(A, i, j) = \bigoplus_{e \in O(v)} \omega(e) \otimes \beta(H(e)) \otimes \bigotimes_{\substack{w \in T(e) \\ w \neq u}} \alpha(w)$$

Outside algorithm

sum over all *outgoing* edges

$$\beta(A, i, j) = \bigoplus_{e \in O(v)} \omega(e) \otimes \beta(H(e)) \otimes \bigotimes_{\substack{w \in T(e) \\ w \neq u}} \alpha(w)$$

Outside algorithm

sum over all *outgoing* edges

$$\beta(A,i,j) = \bigoplus_{e \in O(v)} \omega(e) \otimes \beta(H(e)) \otimes \bigotimes_{\substack{w \in T(e) \\ w \neq u}} \alpha(w)$$

weight of edge

Outside algorithm

sum over all *outgoing* edges

$$\beta(A,i,j) = \bigoplus_{e \in O(v)} \omega(e) \otimes \beta(H(e)) \otimes \bigotimes_{\substack{w \in T(e) \\ w \neq u}} \alpha(w)$$

weight of edge

outside value of node
in the head of the edge

Outside algorithm

sum over all *outgoing* edges

$$\beta(A,i,j) = \bigoplus_{e \in O(v)} \omega(e) \otimes \beta(H(e)) \otimes \bigotimes_{\substack{w \in T(e) \\ w \neq u}} \alpha(w)$$

weight of edge

outside value of node
in the head of the edge

sum over inside values of
the sibling nodes

Inside and Outside algorithms

The final recursions are

$$\alpha(A,i,j) = \omega(A,i,j) \otimes \bigoplus_{k=i+1}^{j-1} \sigma(i,k) \otimes \sigma(k,j)$$

$$\beta(A,i,j) = \bigoplus_{k=1}^{i-1} \sigma'(k,j) \otimes \sigma(k,i) \oplus \bigoplus_{k=j+1}^{n} \sigma'(i,k) \otimes \sigma(j,k),$$

where

$$\sigma(i,j) = \bigoplus_{A \in \Lambda} \alpha(A,i,j),$$

$$\sigma'(i,j) = \bigoplus_{A \in \Lambda} \omega(A,i,j)\beta(A,i,j).$$

References

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics, 2016.

References II

David Gaddy, Mitchell Stern, and Dan Klein. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010. Association for Computational Linguistics, 2018.

Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada, July 2017. Association for Computational Linguistics.