Structured Attention Networks

Yoon Kim*  Carl Denton*  Luong Hoang  Alexander M. Rush

HarvardNLP
1 Deep Neural Networks for Text Processing and Generation

2 Attention Networks

3 Structured Attention Networks
   - Computational Challenges
   - Structured Attention In Practice

4 Conclusion and Future Work
Deep Neural Networks for Text Processing and Generation

Attention Networks

Structured Attention Networks
  • Computational Challenges
  • Structured Attention In Practice

Conclusion and Future Work
Pure Encoder-Decoder Network

Input (sentence, image, etc.)

\[
\text{Encoder}(\text{input}) \in \mathbb{R}^D
\]

Decoder

\[
\text{Decoder}(\text{Encoder}(\text{input}))
\]
Pure Encoder-Decoder Network

Input (sentence, image, etc.)

↓

Fixed-Size Encoder (MLP, RNN, CNN)

Encoder(input) ∈ \( \mathbb{R}^D \)

↓

Decoder

Decoder(Encoder(input))
Pure Encoder-Decoder Network

Input (sentence, image, etc.)

Fixed-Size Encoder (MLP, RNN, CNN)

Encoder(input) ∈ \mathbb{R}^D

Decoder

Decoder(Encoder(input))
Example: Neural Machine Translation (Sutskever et al., 2014)

Over the line!
Example: Neural Machine Translation (Sutskever et al., 2014)

Over the line!
Example: Neural Machine Translation (Sutskever et al., 2014)
Example: Neural Machine Translation (Sutskever et al., 2014)
Example: Neural Machine Translation (Sutskever et al., 2014)
Example: Neural Machine Translation (Sutskever et al., 2014)
Example: Neural Machine Translation (Sutskever et al., 2014)
Example: Neural Machine Translation (Sutskever et al., 2014)
Communication Bottleneck

All input information communicated through fixed-size hidden vector.

Encoder(input)

- Training: All gradients have to flow through single bottleneck.
- Test: All input encoded in single vector.
Neural Attention

Input (sentence, image, etc.)

\[ \text{Encoder(input)} = x_1, x_2, \ldots, x_T \]

Attention Distribution

Annotation Function

“where”

“what”

Context Vector (“soft selection”)

Decoder
Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

Encoder(input) = x_1, x_2, \ldots, x_T

Attention Distribution

Annotation Function

“where”

“what”

Context Vector ("soft selection")

Decoder
Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)
Encoder(input) = \( x_1, x_2, \ldots, x_T \)

Attention Distribution Annotation Function
“where”
“what”

Context Vector (“soft selection”)

Decoder
Neural Attention

Input (sentence, image, etc.)

\[ \text{Encoder}(\text{input}) = x_1, x_2, \ldots, x_T \]

Attention Distribution Annotation Function

“where” “what”

Context Vector (“soft selection”)

Decoder
Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

Encoder(input) = x_1, x_2, \ldots, x_T

Attention Distribution Annotation Function

“where”  “what”

Context Vector ("soft selection")

Decoder
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Attention-based Neural Machine Translation (Bahdanau et al., 2015)
Question Answering (Sukhbaatar et al., 2015)

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

- Greg is a frog
- Brian is a rhino
- Lily is a rhino
- Greg is green
- Brian is white
- John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

- Greg is a frog
- Brian is a rhino
- Lily is a rhino
- Greg is green
- Brian is white
- John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Question Answering (Sukhbaatar et al., 2015)

What color is Lily?

Greg is a frog
Brian is a rhino
Lily is a rhino
Greg is green
Brian is white
John is a frog
Other Applications of Attention Networks

- **Machine Translation** (Bahdanau et al., 2015; Luong et al., 2015)
- **Question Answering** (Hermann et al., 2015; Sukhbaatar et al., 2015)
- **Natural Language Inference** (Rocktäschel et al., 2016; Parikh et al., 2016)
- **Algorithm Learning** (Graves et al., 2014, 2016; Vinyals et al., 2015a)
- **Parsing** (Vinyals et al., 2015b)
- **Speech Recognition** (Chorowski et al., 2015; Chan et al., 2015)
- **Summarization** (Rush et al., 2015)
- **Caption Generation** (Xu et al., 2015)
- and more...
Other Applications: Image Captioning (Xu et al., 2015)

(b) A woman is throwing a frisbee in a park.
Other Applications: Speech Recognition (Chan et al., 2015)
Applications From HarvardNLP: Summarization (Rush et al., 2015)
Applications From HarvardNLP: Image-to-Latex (Deng et al., 2016)

\[ r = \frac{\sqrt{Q_3}}{l} \sin \left( \frac{l}{\sqrt{Q_3}} u \right), \]
1 Deep Neural Networks for Text Processing and Generation

2 Attention Networks

3 Structured Attention Networks
   • Computational Challenges
   • Structured Attention In Practice

4 Conclusion and Future Work
Attention Networks: Notation

\( x_1, \ldots, x_T \) 
Memory bank

\( q \)
Query

\( z \)
Memory selection ("where")

\( p(z = i \mid x, q; \theta) \)
Attention distribution

\( f(x, z) \)
Annotation function ("what")

\( c = \mathbb{E}_z[f(x, z)] \)
Context vector ("soft selection")

End-to-End Requirements:

1. Need to compute attention distribution \( p(z = i \mid x, q; \theta) \)
2. Need to backpropagate to learn parameters \( \theta \)
Attention Networks: Notation

\( x_1, \ldots, x_T \) Memory bank
\( q \) Query
\( z \) Memory selection ("where")
\( p(z = i \mid x, q; \theta) \) Attention distribution
\( f(x, z) \) Annotation function ("what")
\( c = \mathbb{E}_z[f(x, z)] \) Context vector ("soft selection")

End-to-End Requirements:

1. Need to compute attention distribution \( p(z = i \mid x, q; \theta) \)
2. Need to backpropagate to learn parameters \( \theta \)
Attention Networks: Machine Translation

\[ x_1, \ldots, x_T \] Memory bank
\[ q \] Query
\[ z \] Memory selection
\[ p(z = i \mid x, q; \theta) \] Attention distribution
\[ f(x, z) \] Annotation function
\[ c = \mathbb{E}_z[f(x, z)] \] Context vector
Source RNN hidden states
Decoder hidden state
Source position \( \{1, \ldots, T\} \)
softmax(\(x_i^\top q\))
Memory at time \(z\), i.e. \(x_z\)
\[ \sum_{i=1}^T p(z = i \mid x, q)x_i \]

End-to-End Requirements:

1. Need to compute attention \( p(z = i \mid x, q; \theta) \)
   \[ \Rightarrow \] softmax function
2. Need to backpropagate to learn parameters \( \theta \)
   \[ \Rightarrow \] Backprop through softmax function
Attention Networks: Machine Translation

\[ x_1, \ldots, x_T \] Memory bank  \quad Source RNN hidden states

\[ q \] Query  \quad Decoder hidden state

\[ z \] Memory selection  \quad Source position \{1, \ldots, T\}

\[ p(z = i \mid x, q; \theta) \] Attention distribution  \quad \text{softmax}(x_i^T q)

\[ f(x, z) \] Annotation function  \quad Memory at time \( z \), i.e. \( x_z \)

\[ c = \mathbb{E}_z[f(x, z)] \] Context vector  \quad \sum_{i=1}^{T} p(z = i \mid x, q)x_i

End-to-End Requirements:

1. Need to compute attention \( p(z = i \mid x, q; \theta) \)
   \( \implies \) softmax function

2. Need to backpropagate to learn parameters \( \theta \)
   \( \implies \) Backprop through softmax function
Attention Networks: Machine Translation

Over the line! <s>
Attention Networks: Machine Translation

\[ p(z = i \mid x, q) = \text{softmax}(x_i^T q) = \frac{\exp(x_i^T q)}{\sum_{k=1}^{4} \exp(x_k^T q)} \]

\[ p(z = 1 \mid x, q) \quad \ldots \quad p(z = 4 \mid x, q) \]

\( x_1 \quad x_2 \quad x_3 \quad x_4 \quad q \)

Over the line ! <s>
Attention Networks: Machine Translation

\[ c = \sum_{i=1}^{4} p(z = i \mid x, q) x_i = \mathbb{E}_{z \sim p(z \mid x, q)}[x_z] \]
Attention Networks: Machine Translation

\[ p(w_1 | x) = \text{softmax}(\text{MLP}([c; q])) \]
1 Deep Neural Networks for Text Processing and Generation

2 Attention Networks

3 Structured Attention Networks
   - Computational Challenges
   - Structured Attention In Practice

4 Conclusion and Future Work
Structured Attention Networks

- Replace simple attention with distribution over a combinatorial set of structures
- Attention distribution represented with graphical model over multiple latent variables
- Compute attention using embedded inference

New Model

\[ p(z \mid x, q; \theta) \quad \text{Attention distribution over structures } z \]
Structured Attention Networks: Notation

\[ x_1, \ldots, x_T \quad \text{Memory bank} \]
\[ q \quad \text{Query} \]
\[ z = z_1, \ldots, z_T \quad \text{Memory selection over structures} \]
\[ p(z \mid x, q; \theta) \quad \text{Attention distribution over structures} \]
\[ f(x, z) \quad \text{Annotation function (Neural representation)} \]
\[ c = \mathbb{E}_{z \sim p(z \mid x, q)}[f(x, z)] \quad \text{Context vector} \]

Consider family of functions \( f(x, z) \) that makes \( \mathbb{E}_{z \sim p(z \mid x, q)}[f(x, z)] \) computationally tractable.
Structured Attention Networks: Notation

\[ x_1, \ldots, x_T \quad \text{Memory bank} \]
\[ q \quad \text{Query} \]
\[ z = z_1, \ldots, z_T \quad \text{Memory selection over structures} \]
\[ p(z | x, q; \theta) \quad \text{Attention distribution over structures} \]
\[ f(x, z) \quad \text{Annotation function (Neural representation)} \]
\[ c = \mathbb{E}_{z \sim p(z | x, q)}[f(x, z)] \quad \text{Context vector} \]

Consider family of functions \( f(x, z) \) that makes \( \mathbb{E}_{z \sim p(z | x, q)}[f(x, z)] \)
computationally tractable
Structured Attention Networks for Neural Machine Translation

\[
\sum_{i=1}^{4} p(z = i \mid x, q) = 1
\]

\(x_1\) \(x_2\) \(x_3\) \(x_4\)

Over the line ! <s>
Structured Attention Networks for Neural Machine Translation

\[ x_1 \rightarrow 0.1 \rightarrow x_2 \rightarrow 0.9 \rightarrow x_3 \rightarrow 0.9 \rightarrow x_4 \rightarrow 0.1 \rightarrow q \]

Over the line! <s>
Structured Attention Networks for Neural Machine Translation

\[ p(z_1 = 1 \mid x, q) = \text{sigmoid}(x_1^\top q) \]
\[ \cdots \]

\[ p(z_4 = 1 \mid x, q) = \text{sigmoid}(x_4^\top q) \]

- Over
- the
- line
- !
- <s>
Structured Attention Networks for Neural Machine Translation

\[ p(z_1, z_2, z_3, z_4 \mid x, q) = \prod_{i=1}^{4} p(z_i \mid x, q) \]

\[
\begin{align*}
z_i &= 1 & \\
z_i &= 0
\end{align*}
\]

\[
\begin{align*}
x_1 & \quad x_2 & \quad x_3 & \quad x_4
\end{align*}
\]

\[
\begin{align*}
\text{Over} & \quad \text{the} & \quad \text{line} & \quad !
\end{align*}
\]

\[
\begin{align*}
q & <s> \quad <s>
\end{align*}
\]
Structured Attention Networks for Neural Machine Translation

\[ p(z_1, z_2, z_3, z_4 \mid x, q) = \text{softmax}\{\theta(z_1, z_2, z_3, z_4)\} \]
\[ = \frac{1}{Z} \exp(\theta(z_1, z_2, z_3, z_4)) \]

\( z_i = 1 \)
\( z_i = 0 \)

Over \( x_1 \) \( x_2 \) \( x_3 \) \( x_4 \) \( q \) <s>
Structured Attention Networks for Neural Machine Translation

\[ p(z_1, z_2, z_3, z_4 \mid x, q) = \text{softmax}(\theta(z_1, z_2, z_3, z_4)) \]
\[ = \frac{1}{Z} \exp(\theta(z_1, z_2, z_3, z_4)) \]

\[ Z = \sum_{[z'_1, z'_2, z'_3, z'_4] \in \{0,1\}^4} \exp(\theta(z'_1, z'_2, z'_3, z'_4)) \]
Structured Attention Networks for Neural Machine Translation

\[ p(z_1 = 0, z_2 = 1, z_3 = 1, z_4 = 0 \mid x, q) \]

\[ z_i = 1 \]

\[ z_i = 0 \]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]

\[ \text{Over} \quad \text{the} \quad \text{line} \quad ! \]

\[ q \]

\[ <s> \]
Structured Attention Networks for Neural Machine Translation

\[ p(z_1 = 0, z_2 = 0, z_3 = 1, z_4 = 0 \mid x, q) \]

\[ z_i = 1 \quad \square \quad \square \quad \square \quad \square \\
\[ z_i = 0 \quad \square \quad \square \quad \square \quad \square \\
\]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]

Over the line !

q <s>
Structured Attention Networks for Neural Machine Translation

\[ c = \sum_{z_1, z_2, z_3, z_4} p(z_1, z_2, z_3, z_4 \mid x, q)f(x, z) = \mathbb{E}_{z \sim p(z \mid x, q)}[f(x, z)] = \]

\[ z_i = 1 \quad \text{or} \quad z_i = 0 \]

\[ \begin{align*}
  & x_1 & x_2 & x_3 & x_4 \\
  \text{Over} & \quad \text{the} & \quad \text{line} & \quad ! \\
  q & \quad <s> \end{align*} \]
Structured Attention Networks for Neural Machine Translation

\[ f(x, z) = \sum_{i=1}^{4} \mathbb{1}\{z_i = 1\}x_i \]

\[ \mathbb{E}_{z \sim p(z|x, q)}[f(x, z)] = \sum_{i=1}^{4} p(z_i = 1 \mid x, q)x_i \]
Motivation: Structured Output Prediction

Modeling the structured output (i.e. graphical model on top of a neural net) has improved performance (LeCun et al., 1998; Lafferty et al., 2001; Collobert et al., 2011)

- Given a sequence $x = x_1, \ldots, x_T$
- Factored potentials $\theta_{i,i+1}(z_i, z_{i+1}; x)$

\[
p(z_1 \ldots, z_T | x; \theta) = \text{softmax} \left( \sum_{i=1}^{T-1} \theta_{i,i+1}(z_i, z_{i+1}; x) \right) \]

\[
= \frac{1}{Z} \exp \left( \sum_{i=1}^{T-1} \theta_{i,i+1}(z_i, z_{i+1}; x) \right) \]

\[
Z = \sum_{z' \in C} \exp \left( \sum_{i=1}^{T-1} \theta_{i,i+1}(z'_i, z'_{i+1}; x) \right) \]
Example: Part-of-Speech Tagging

christian  bale  is  the  best  batman

NNP: proper noun
NN: noun
JJ: adjective
DT: determiner
VBZ: verb
**Example: Part-of-Speech Tagging**

<table>
<thead>
<tr>
<th>NNP</th>
<th>NNP</th>
<th>VBZ</th>
<th>DT</th>
<th>JJ</th>
<th>NNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>christian</td>
<td>bale</td>
<td>is</td>
<td>the</td>
<td>best</td>
<td>batman</td>
</tr>
</tbody>
</table>
Example: Part-of-Speech Tagging

NNP: proper noun
NN: noun
JJ: adjective
DT: determiner
VBZ: verb

christian    bale    is    the    best    batman
Example: Part-of-Speech Tagging

- JJ: adjective
- NN: noun
- VBZ: verb
- DT: determiner
- NNP: proper noun

- CHRISTIAN
- bale
- is
- the
- best
- batman
Example: Part-of-Speech Tagging

- **JJ**: adjective
- **NN**: noun
- **VBZ**: verb
- **DT**: determiner

Examples:

- *christian* (JJ)
- *bale* (NN)
- *is* (VBZ)
- *the* (DT)
- *best* (JJ)
- *batman* (NNP)
Neural CRF for Sequence Tagging (Collobert et al., 2011)

- NNP: proper noun
- NN: noun
- JJ: adjective
- DT: determiner
- VBZ: verb

christian → bale → is → the → best → batman
Neural CRF for Sequence Tagging (Collobert et al., 2011)

Unary potentials $\theta_i(c) = w_c^T x_i$ come from neural network
Inference in Linear-Chain CRF

Pairwise potentials are simple parameters $b$, so altogether

$$\theta_{i,i+1}(c, d) = \theta_i(c) + \theta_{i+1}(d) + b_{c,d}$$
1 Deep Neural Networks for Text Processing and Generation

2 Attention Networks

3 Structured Attention Networks
   - Computational Challenges
   - Structured Attention In Practice

4 Conclusion and Future Work
Structured Attention Networks: Notation

\[ x_1, \ldots, x_T \quad \text{Memory bank} \]
\[ q \quad \text{Query} \]
\[ z = z_1, \ldots, z_T \quad \text{Memory selection over structures} \]
\[ p(z \mid x, q; \theta) \quad \text{Attention distribution over structures} \]
\[ f(x, z) \quad \text{Annotation function (Neural representation)} \]
\[ c = \mathbb{E}_{z \sim p(z \mid x, q)}[f(x, z)] \quad \text{Context vector} \]

Need to calculate

\[ c = \sum_{i=1}^{T} p(z_i = 1 \mid x, q)x_i \]
Challenge: End-to-End Training

Requirements:

1. Compute attention distribution (marginals) \( p(z_i \mid x, q; \theta) \)
   \[ \implies \text{Forward-backward algorithm} \]

2. Gradients wrt attention distribution parameters \( \theta \)
   \[ \implies \text{Backpropagation through forward-backward algorithm} \]
Challenge: End-to-End Training

Requirements:

1. Compute attention distribution (marginals) $p(z_i | x, q; \theta)$
   $\implies$ Forward-backward algorithm

2. Gradients wrt attention distribution parameters $\theta$
   $\implies$ Backpropagation through forward-backward algorithm
Challenge: End-to-End Training

Requirements:

1. Compute attention distribution (marginals) $p(z_i | x, q; \theta)$
   $\implies$ Forward-backward algorithm

2. Gradients wrt attention distribution parameters $\theta$
   $\implies$ Backpropagation through forward-backward algorithm
Review: Forward-Backward Algorithm

$\theta$: input potentials (e.g. from NN)
$\alpha, \beta$: dynamic programming tables

**procedure** \textsc{ForwardBackward($\theta$)}

**Forward**

for $i = 1, \ldots, n$; $z_i$ do

$\alpha[i, z_i] \leftarrow \sum_{z_{i-1}} \alpha[i - 1, z_{i-1}] \times \exp(\theta_{i-1,i}(z_{i-1}, z_i))$

**Backward**

for $i = n, \ldots, 1$; $z_i$ do

$\beta[i, z_i] \leftarrow \sum_{z_{i+1}} \beta[i + 1, z_{i+1}] \times \exp(\theta_{i,i+1}(z_i, z_{i+1}))$

**Marginals**

for $i = 1, \ldots, n$; $c \in C$ do

$p(z_i = c \mid x) \leftarrow \alpha[i, c] \times \beta[i, c] / Z$
Forward-Backward Algorithm in Practice (Log-Space Semiring Trick)

\[ x \oplus y = \log(\exp(x) + \exp(y)) \]
\[ x \otimes y = x + y \]

**procedure** ForwardBackward(\( \theta \))

**Forward**

\[ \text{for } i = 1, \ldots, n; \; z_i \text{ do} \]
\[ \alpha[i, z_i] \leftarrow \bigoplus_{z_{i-1}} \alpha[i - 1, y] \otimes \theta_{i-1,i}(z_{i-1}, z_i) \]

**Backward**

\[ \text{for } i = n, \ldots, 1; \; z_i \text{ do} \]
\[ \beta[i, z_i] \leftarrow \bigoplus_{z_{i+1}} \beta[i + 1, z_{i+1}] \otimes \theta_{i,i+1}(z_i, z_{i+1}) \]

**Marginals**

\[ \text{for } i = 1, \ldots, n; \; c \in \mathcal{C} \text{ do} \]
\[ p(z_i = c \mid x) \leftarrow \exp(\alpha[i, c] \otimes \beta[i, c] \otimes -\log Z) \]
Backpropagating through Forward-Backward

$$\nabla_\mathcal{L}_p: \text{ Gradient of arbitrary loss } \mathcal{L} \text{ with respect to marginals } p$$

**procedure** `BACKPROPFORWARDBACKWARD(\theta, p, \nabla_\mathcal{L}_p)`

**Backprop Backward**

```plaintext
for i = n, \ldots, 1; z_i do
    \hat{\beta}[i, z_i] ← \nabla_\mathcal{L}_\alpha[i, z_i] \oplus \bigoplus_{z_{i+1}} \theta_{i,i+1}(z_i, z_{i+1}) \otimes \hat{\beta}[i + 1, z_{i+1}]
```

**Backprop Forward**

```plaintext
for i = 1, \ldots, n; z_i do
    \hat{\alpha}[i, z_i] ← \nabla_\mathcal{L}_\beta[i, z_i] \oplus \bigoplus_{z_{i-1}} \theta_{i-1,i}(z_{i-1}, z_i) \otimes \hat{\alpha}[i - 1, z_{i-1}]
```

**Potential Gradients**

```plaintext
for i = 1, \ldots, n; z_i, z_{i+1} do
    \nabla_{\theta_{i-1,i}} \mathcal{L}(z_i, z_{i+1}) ← \exp(\hat{\alpha}[i, z_i] \otimes \beta[i + 1, z_{i+1}] \oplus \alpha[i, z_i] \otimes \hat{\beta}[i + 1, z_{i+1}] \oplus \alpha[i, z_i] \otimes \beta[i + 1, z_{i+1}] \otimes -\log Z)
```
Interesting Issue: Negative Gradients Through Attention

- $\nabla_p^L$: Gradient could be negative, but working in log-space!
- Signed Log-space semifield trick (Li and Eisner, 2009)
- Use tuples $(l_a, s_a)$ where $l_a = \log |a|$ and $s_a = \text{sign}(a)$

\[ \begin{array}{c|c|c|c|}
  & & \oplus & \\
  s_a & s_b & l_{a+b} & s_{a+b} \\
  \hline
  + & + & l_a + \log(1 + d) & + \\
  + & - & l_a + \log(1 - d) & + \\
  - & + & l_a + \log(1 - d) & - \\
  - & - & l_a + \log(1 + d) & - \\
\end{array} \]

(Similar rules for $\otimes$)
Structured Attention Networks for Neural Machine Translation

\[ \frac{\partial L}{\partial p_1} \quad \frac{\partial L}{\partial p_2} \quad \frac{\partial L}{\partial p_3} \quad \frac{\partial L}{\partial p_4} \quad \frac{\partial L}{\partial c} \]

BackpropForwardBackward(\( \frac{\partial L}{\partial \theta} \))

\[ z_i = 1 \quad z_i = 0 \]

Over the line!

<s>
1 Deep Neural Networks for Text Processing and Generation

2 Attention Networks

3 Structured Attention Networks
   - Computational Challenges
   - Structured Attention In Practice

4 Conclusion and Future Work
Implementation

http://github.com/harvardnlp/struct-attn

- General-purpose structured attention unit
- “Plug-and-play” neural network layers
- Dynamic programming is GPU-optimized for speed
NLP Experiments

Replace existing attention layers for

- **Machine Translation**
  - **Segmental Attention**: 2-state linear-chain CRF
- **Question Answering**
  - **Sequential Attention**: $N$-state linear-chain CRF
- **Natural Language Inference**
  - **Syntactic Attention**: graph-based dependency parser
Segmental Attention for Neural Machine Translation

- Use segmentation CRF for attention, i.e. binary vectors of length $n$
- $p(z_1, \ldots, z_T \mid x, q)$ parameterized with a linear-chain CRF.

Unary potentials (Encoder RNN):

$$\theta_i(k) = \begin{cases} 
  x_i W q, & k = 1 \\
  0, & k = 0 
\end{cases}$$

Pairwise potentials (Simple Parameters):

4 additional binary parameters (i.e., $b_{0,0}, b_{0,1}, b_{1,0}, b_{1,1}$)
Segmental Attention for Neural Machine Translation

Data:

- Japanese → English (from WAT 2015)
- Traditionally, word segmentation as a preprocessing step
- Use structured attention learn an implicit segmentation model

Experiments:

- Japanese characters → English words
- Japanese words → English words
### Segmental Attention for Neural Machine Translation

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Sigmoid</th>
<th>Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHAR → WORD</strong></td>
<td>12.6</td>
<td>13.1</td>
<td>14.6</td>
</tr>
<tr>
<td><strong>WORD → WORD</strong></td>
<td>14.1</td>
<td>13.8</td>
<td>14.3</td>
</tr>
</tbody>
</table>

**BLEU scores on test set (higher is better)**

- **Models:**
  - Simple softmax attention: $\text{softmax}(\theta_i)$
  - Sigmoid attention: $\text{sigmoid}(\theta_i)$
  - Structured attention: $\text{ForwardBackward}(\theta)$
Attention Visualization: Ground Truth

There were two problems in the solution of the electrification problem.
Attention Visualization: Simple Attention

<table>
<thead>
<tr>
<th>帯電問題の解決には二つの課題があった。</th>
</tr>
</thead>
<tbody>
<tr>
<td>There were two problems in solution</td>
</tr>
<tr>
<td>of the electrification problem</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Attention Visualization: Sigmoid Attention

<table>
<thead>
<tr>
<th>带 電 問 題 の 解 決 に は 二 つ の 課 題 が あ っ た。</th>
</tr>
</thead>
<tbody>
<tr>
<td>There were two problems in the solution of the electrification problem.</td>
</tr>
</tbody>
</table>
Attention Visualization: Structured Attention

There were two problems in the solution of electrification problems.
Sequential Attention over Facts for Question Answering

Simple attention: Greedy soft-selection of \( K \) supporting facts
Sequential Attention over Facts for Question Answering

Structured attention: Consider all possible sequences

Lily is a rhino  →  Brian is a rhino  →  Brian is white

- Greg is a frog
- Brian is a rhino
- Lily is a rhino
- Greg is green
- Brian is white
- John is a frog
### Sequential Attention over Facts for Question Answering

**BaBi tasks (Weston et al., 2015): 1k questions per task**

<table>
<thead>
<tr>
<th>Task</th>
<th>(K)</th>
<th>Simple</th>
<th>Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ans %</td>
<td>Fact %</td>
</tr>
<tr>
<td>Task 02</td>
<td>2</td>
<td>87.3</td>
<td>46.8</td>
</tr>
<tr>
<td>Task 03</td>
<td>3</td>
<td>52.6</td>
<td>1.4</td>
</tr>
<tr>
<td>Task 11</td>
<td>2</td>
<td>97.8</td>
<td>38.2</td>
</tr>
<tr>
<td>Task 13</td>
<td>2</td>
<td>95.6</td>
<td>14.8</td>
</tr>
<tr>
<td>Task 14</td>
<td>2</td>
<td>99.9</td>
<td>77.6</td>
</tr>
<tr>
<td>Task 15</td>
<td>2</td>
<td>100.0</td>
<td>59.3</td>
</tr>
<tr>
<td>Task 16</td>
<td>3</td>
<td>97.1</td>
<td>91.0</td>
</tr>
<tr>
<td>Task 17</td>
<td>2</td>
<td>61.1</td>
<td>23.9</td>
</tr>
<tr>
<td>Task 18</td>
<td>2</td>
<td>86.4</td>
<td>3.3</td>
</tr>
<tr>
<td>Task 19</td>
<td>2</td>
<td>21.3</td>
<td>10.2</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>–</td>
<td>81.4</td>
<td>39.6</td>
</tr>
</tbody>
</table>
**Question**: what color is bernhard? green

**Correct Facts**: 5, 6, 8
Natural Language Inference

Given a premise (P) and a hypothesis (H), predict the relationship: Entailment (E), Contradiction (C), Neutral (N)

<table>
<thead>
<tr>
<th>P</th>
<th>The boy is running through a grassy area.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The boy is in his room.</td>
<td>C</td>
</tr>
<tr>
<td>A boy is running outside.</td>
<td>E</td>
</tr>
<tr>
<td>The boy is in a park.</td>
<td>N</td>
</tr>
</tbody>
</table>

Many existing models run parsing as a preprocessing step and attend over parse trees.
Natural Language Inference

Given a premise (P) and a hypothesis (H), predict the relationship:
Entailment (E), Contradiction (C), Neutral (N)

<table>
<thead>
<tr>
<th>P</th>
<th>The boy is running through a grassy area.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The boy is in his room.</td>
</tr>
<tr>
<td>H</td>
<td>A boy is running outside.</td>
</tr>
<tr>
<td></td>
<td>The boy is in a park.</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>

Many existing models run parsing as a preprocessing step and attend over parse trees.
Neural CRF Parsing (Durrett and Klein, 2015; Kipperwasser and Goldberg, 2016)

\[ z_{ij} = 1 \implies \text{i-th word is parent of j-th word} \]
Neural CRF Parsing (Durrett and Klein, 2015; Kipperwasser and Goldberg, 2016)

\[ z_{1,3} = 1 \]
\[ z_{3,2} = 1 \]
\[ z_{3,5} = 1 \]
\[ z_{5,4} = 1 \]
Syntactic Attention Network

1. Attention distribution (probability of a parse tree)
   \[ \Rightarrow \text{Inside/outside algorithm} \]

2. Gradients wrt attention distribution parameters: \( \frac{\partial L}{\partial \theta} \)
   \[ \Rightarrow \text{Backpropagation through inside/outside algorithm} \]

Forward/backward pass on inside-outside version of Eisner’s algorithm (Eisner, 1996) takes \( O(T^3) \) time.
Syntactic Attention Network

1. **Attention distribution** (probability of a parse tree)  
   $\Rightarrow$ **Inside/outside algorithm**

2. **Gradients wrt attention distribution parameters:** $\frac{\partial L}{\partial \theta}$  
   $\Rightarrow$ **Backpropagation through inside/outside algorithm**

Forward/backward pass on inside-outside version of Eisner’s algorithm (Eisner, 1996) takes $O(T^3)$ time.
Syntactic Attention Network

1. Attention distribution (probability of a parse tree) $
   \implies$ Inside/outside algorithm

2. Gradients wrt attention distribution parameters: $\frac{\partial L}{\partial \theta}$
   $
   \implies$ Backpropagation through inside/outside algorithm

Forward/backward pass on inside-outside version of Eisner’s algorithm (Eisner, 1996) takes $O(T^3)$ time.
Forward/Back-propagation through Inside-Outside Algorithm

**Procedure InsideOutside()**

```
α, β ← ∞
for i = 1, ..., n do
  α[i, t, L, 1] ← 0
  β[i, t, R, 1] ← 0
β[1, n, R, 1] ← 0
```

**Inside step**

```
for k = 1, ..., n do
  for s = 1, ..., n - k do
    t ← s + k
    α[s, t, R, 0] ← θ[s, t, R, 0] α[s, t, R, 1] + α[s + 1, t, L, 1] + θ[s + 1, t, R, 0] + θ[s, t, L, 0]
    β[s, t, L, 1] ← θ[s, t, L, 1] β[s, t, L, 0] + β[s + 1, t, R, 1] + θ[s + 1, t, L, 0]
```

**Outside step**

```
for k = n, ..., 1 do
  for s = 1, ..., n - k do
    u ← s + k
    β[s, u, R, 0] ← β[s, u, R, 1] + β[u + 1, t, R, 1] + θ[u + 1, t, R, 0] + θ[s, t, L, 0]
    α[s, u, L, 1] ← α[s, u, L, 0] + α[s, t, L, 0] + β[u + 1, t, R, 1] + θ[u + 1, t, R, 0] + θ[s, t, L, 0]
```

**Log partition**

```
A ← α[1, n, L, 1] + β[1, n, R, 1]
```

**Compute marginals**

```
p[s, t] ← exp(α[s, t, R, 0] + β[s, t, L, 0] - A)
```

**Return**

```
return p
```

**Procedure BackpropInsideOutside()**

```
γ ← ∞
for s = 1, ..., n do
  γ[s, t, R, 0] ← γ[s, t, R, 1] + γ[s, t, L, 0] + γ[s, t, L, 1] + γ[s + 1, t, R, 0] + γ[s + 1, t, L, 0] + γ[s, t, L, 0]
```

**Backpropagate through outside step**

```
for s = 1, ..., n - k do
  t ← s + k
  γ[s, t, R, 0] ← γ[s, t, R, 1] + γ[s, t, L, 0] + γ[s, t, L, 1] + γ[s + 1, t, R, 0] + γ[s + 1, t, L, 0] + γ[s, t, L, 0]
```

**Backpropagate through inside step**

```
for k = n, ..., 1 do
  for s = 1, ..., n - k do
    u ← s + k
    γ[s, u, R, 0] ← γ[s, u, R, 1] + γ[s, u, L, 0] + γ[s, u, L, 1] + γ[s + 1, u, R, 0] + γ[s + 1, u, L, 0] + γ[s, u, L, 0]
```

**Exponentiate log gradient, multiply by sign, and return ∇F**
Syntactic Attention

\[ p(z_{ij} | x) \quad \forall i \neq j \]
Syntactic Attention
Syntactic Attention
Syntactic Attention

\[ c_2 = \text{soft-parent}(\text{John}) = \sum_{i=1}^{5} p(z_{i,2} = 1 \mid x)x_i \]
Syntactic Attention

\[ c_3 = \text{soft-parent}(\text{hit}) \]
\[ = \sum_{i=1}^{5} p(z_{i,3} = 1 | x) x_i \]
Syntactic Attention

\[ c_4 = \text{soft-parent}(\text{the}) = \sum_{i=1}^{5} p(z_{i,4} = 1 | x) x_i \]
Syntactic Attention

\[ c_5 = \text{soft-parent}(\text{ball}) = \sum_{i=1}^{5} p(z_{i,5} = 1 | x) x_i \]
Syntactic Attention for Natural Language Inference

Dataset: Stanford Natural Language Inference (Bowman et al., 2015)

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Attention</td>
<td>85.8</td>
</tr>
<tr>
<td>Hard parent</td>
<td>86.1</td>
</tr>
<tr>
<td>Simple Attention</td>
<td>86.2</td>
</tr>
<tr>
<td>Structured Attention</td>
<td>86.8</td>
</tr>
</tbody>
</table>

- No attention: word embeddings only
- “Hard” parent from a pipelined dependency parser
- Simple attention (simple softmax instead of syntactic attention)
- Structured attention (soft parents from syntactic attention)
Syntactic Attention for Natural Language Inference

Run Viterbi algorithm on the parsing layer to get the MAP parse:

$$\hat{z} = \arg \max_z p(z | x, q)$$

The men are fighting outside a deli.
1. Deep Neural Networks for Text Processing and Generation

2. Attention Networks

3. Structured Attention Networks
   - Computational Challenges
   - Structured Attention In Practice

4. Conclusion and Future Work
Structured Attention Networks

- Generalize attention to incorporate latent structure
- Exact inference through dynamic programming
- Training remains end-to-end

Future work

- Approximate differentiable inference in neural networks
- Incorporate other probabilistic models into deep learning
- Compare further to methods using EM or hard structures


References V


