Structured Attention Networks

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

Harvard University

Summary

- Generalize attention to incorporate latent input structure
- Attention over a combinatorial set (segmentations, parse trees) $\implies$ exact inference via dynamic programming
- Backpropagate **through** inference $\implies$ training remains end-to-end
- Attention layers learn "segmenter/parser as a hidden layer"

Attention Networks

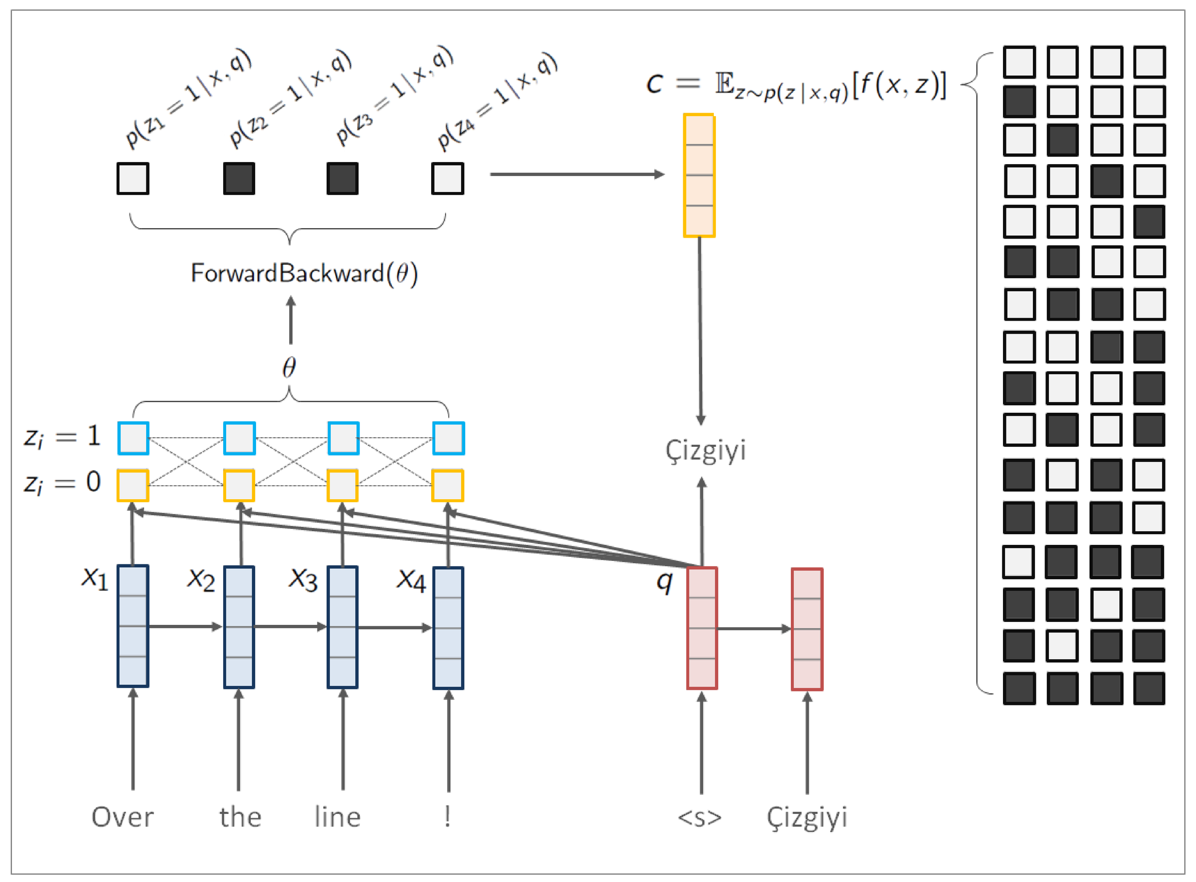
- Attention networks: represent input with a variable length **memory bank** (instead of a fixed-dimensional vector).
- Requires computing **attention distribution** ("where") and **annotation function** ("what") given source and query.
- **Context vector** is defined as the expected annotation.

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

Structured Attention Networks

- Input structure represented with a **graphical model** over multiple latent variables $z = z_1, \ldots, z_m$.
- $p(z \mid x, q; \theta)$: attention distribution over set of **structures**.
- Compute attention/context vector using **embedded inference**.

Segmental Attention: Machine Translation



- Latent variables: $z = [z_1, \ldots, z_T]$ (binary vectors of length T)
- Attention distribution from a 2-state, linear-chain CRF:
  - Unary potentials: $\theta_i(k) = x_i^\top W q$, if $k = 1$ (otherwise 0)
  - Pairwise potentials: binary parameters (i.e., $b_{0,0}, b_{0,1}, b_{1,0}, b_{1,1}$)
- Annotation function: $f(x, z) = \sum_{i=1}^T \mathbb{1}\{z_i = 1\} x_i$
- Context vector: $c = \mathbb{E}_z[f(x, z)] = \sum_{i=1}^T p(z_i = 1 \mid x, q; \theta) x_i$

Backpropagation through Inference

procedure FORWARDBACKWARD(θ)
- $\alpha \leftarrow 0$
- $\beta \leftarrow 0$
- Forward
- **for** $i = 1, \ldots, n$; z_i **do**
  - $\alpha[i, z_i] \leftarrow \oplus_{z_{i-1}} \alpha[i-1, y] \otimes \theta_{i-1,i}(z_{i-1}, z_i)$
- Backward
- **for** $i = n, \ldots, 1$; z_i **do**
  - $\beta[i, z_i] \leftarrow \oplus_{z_{i+1}} \beta[i+1, z_{i+1}] \otimes \theta_{i,i+1}(z_i, z_{i+1})$
- $\log Z \leftarrow \oplus_{c \in \mathcal{C}} \alpha[n, c]$
- Marginals
- **for** $i = 1, \ldots, n$; $c \in \mathcal{C}$ **do**
  - $p(z_i = c \mid x) \leftarrow \exp(\alpha[i, c] \otimes \beta[i, c] \otimes -\log Z)$
- **return** p

procedure BACKPROPFORWARDBACKWARD($\theta, p, \nabla_p^{\mathcal{L}}$)
- $\nabla_\alpha^{\mathcal{L}} \leftarrow \log p \otimes \log \nabla_p^{\mathcal{L}} \otimes \beta \otimes -\log Z$
- $\nabla_\beta^{\mathcal{L}} \leftarrow \log p \otimes \log \nabla_p^{\mathcal{L}} \otimes \alpha \otimes -\log Z$
- Backprop Backward
- **for** $i = n, \ldots 1$; z_i **do**
  - $\hat{\beta}[i, z_i] \leftarrow \nabla_\alpha^{\mathcal{L}}[i, z_i] \oplus \bigoplus_{z_{i+1}} \theta_{i,i+1}(z_i, z_{i+1}) \otimes \hat{\beta}[i+1, z_{i+1}]$
- Backprop Forward
- **for** $i = 1, \ldots, n$; z_i **do**
  - $\hat{\alpha}[i, z_i] \leftarrow \nabla_\beta^{\mathcal{L}}[i, z_i] \oplus \bigoplus_{z_{i-1}} \theta_{i-1,i}(z_{i-1}, z_i) \otimes \hat{\alpha}[i-1, z_{i-1}]$
- Potential Gradients
- **for** $i = 1, \ldots, n$; z_i, z_{i+1} **do**
  - $\nabla_{\theta_{i-1,i}(z_i, z_{i+1})}^{\mathcal{L}} \leftarrow \exp(\hat{\alpha}[i, z_i] \otimes \beta[i+1, z_{i+1}] \oplus \alpha[i, z_i] \otimes \hat{\beta}[i+1, z_{i+1}] \oplus \alpha[i, z_i] \otimes \beta[i+1, z_{i+1}] \otimes -\log Z)$
- **return** $\nabla_\theta^{\mathcal{L}}$

Signed Log-Space Semifield

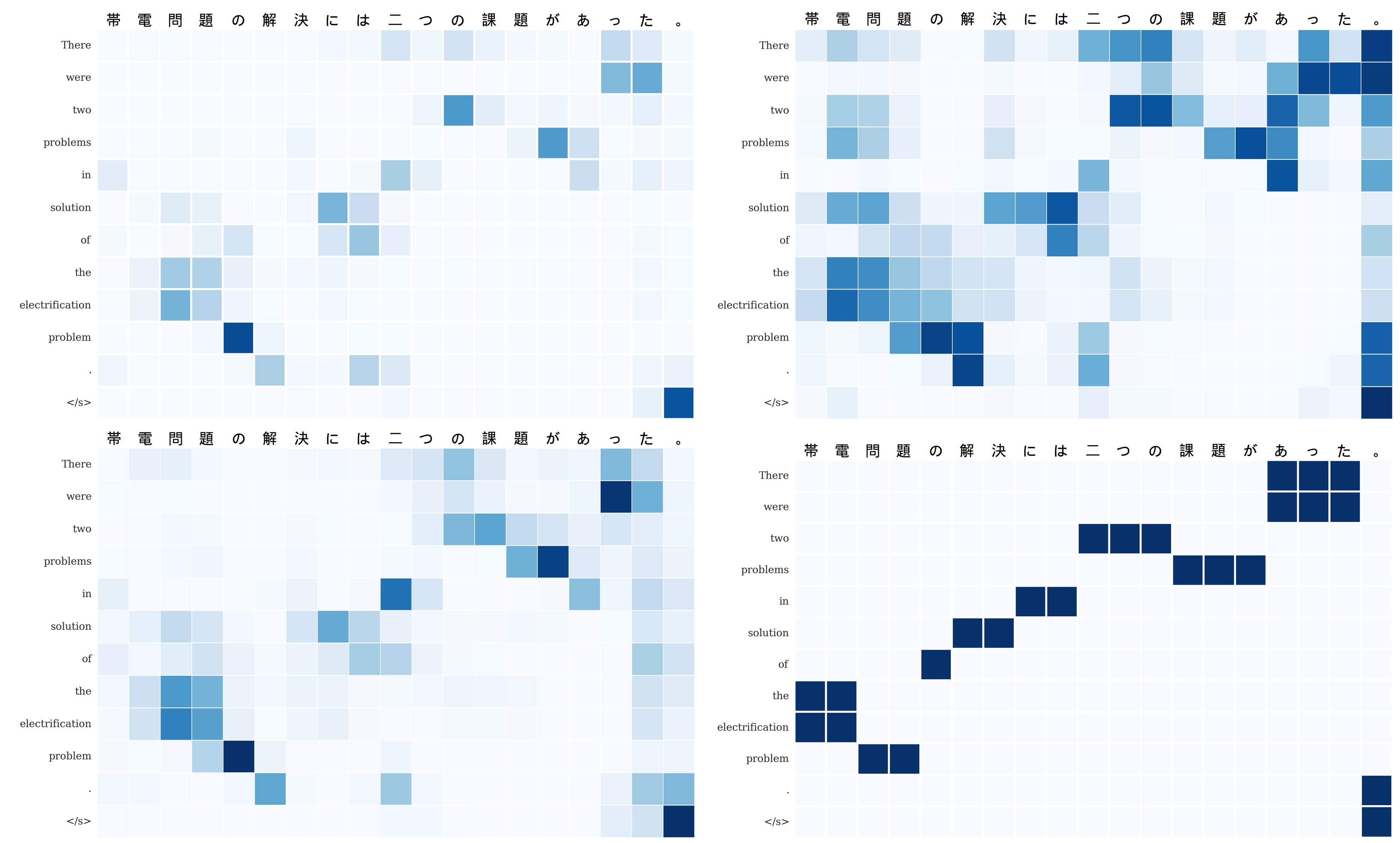
- FORWARDBACKWARD(θ) $\rightarrow p$: a differentiable function that takes in potentials θ and produces marginals p.
- Forward-prop: need to work in (standard) log-space semiring with ($\oplus = \mathrm{logadd}, \otimes = +$) to avoid numerical underflow.
- Backprop: want to backprop $\nabla_p^{\mathcal{L}}$ while in log-space, but $\nabla_p^{\mathcal{L}}$ could be negative!
- Need to extend to the **signed** log-space semifield where a is represented as a tuple ($l_a = \log|a|$, $s_a = \mathrm{sign}(a)$) with modified binary operations $\oplus, \otimes$.

		$\oplus$		$\otimes$	
s_a	s_b	l_{a+b}	s_{a+b}	$l_{a \cdot b}$	$s_{a \cdot b}$
$+$	$+$	$l_a + \log(1 + d)$	$+$	$l_a + l_b$	$+$
$+$	$-$	$l_a + \log(1 - d)$	$+$	$l_a + l_b$	$-$
$-$	$+$	$l_a + \log(1 - d)$	$-$	$l_a + l_b$	$-$
$-$	$-$	$l_a + \log(1 + d)$	$-$	$l_a + l_b$	$+$

Signed log-space semifield (Li & Eisner 2009). Here we assume $d = \exp(l_b - l_a)$ and $|a| > |b|$.

- Simple: $p(z = i \mid x, q) = \mathrm{softmax}(\theta_i)$
- Sigmoid: $p(z_i = 1 \mid x, q) = \mathrm{sigmoid}(\theta_i)$
- Structured: $p(z_i = 1 \mid x, q) = $ FORWARDBACKWARD(θ)

BLEU	Simple	Sigmoid	Structured
Char $\rightarrow$ Word	12.6	13.1	14.6
Word $\rightarrow$ Word	14.1	13.8	14.3



Visualization of the source attention distribution for the simple (top left), sigmoid (top right), and structured (bottom left) attention models over a ground truth sentence on the character-to-word translation task. Bottom right shows the manually-annotated alignments.

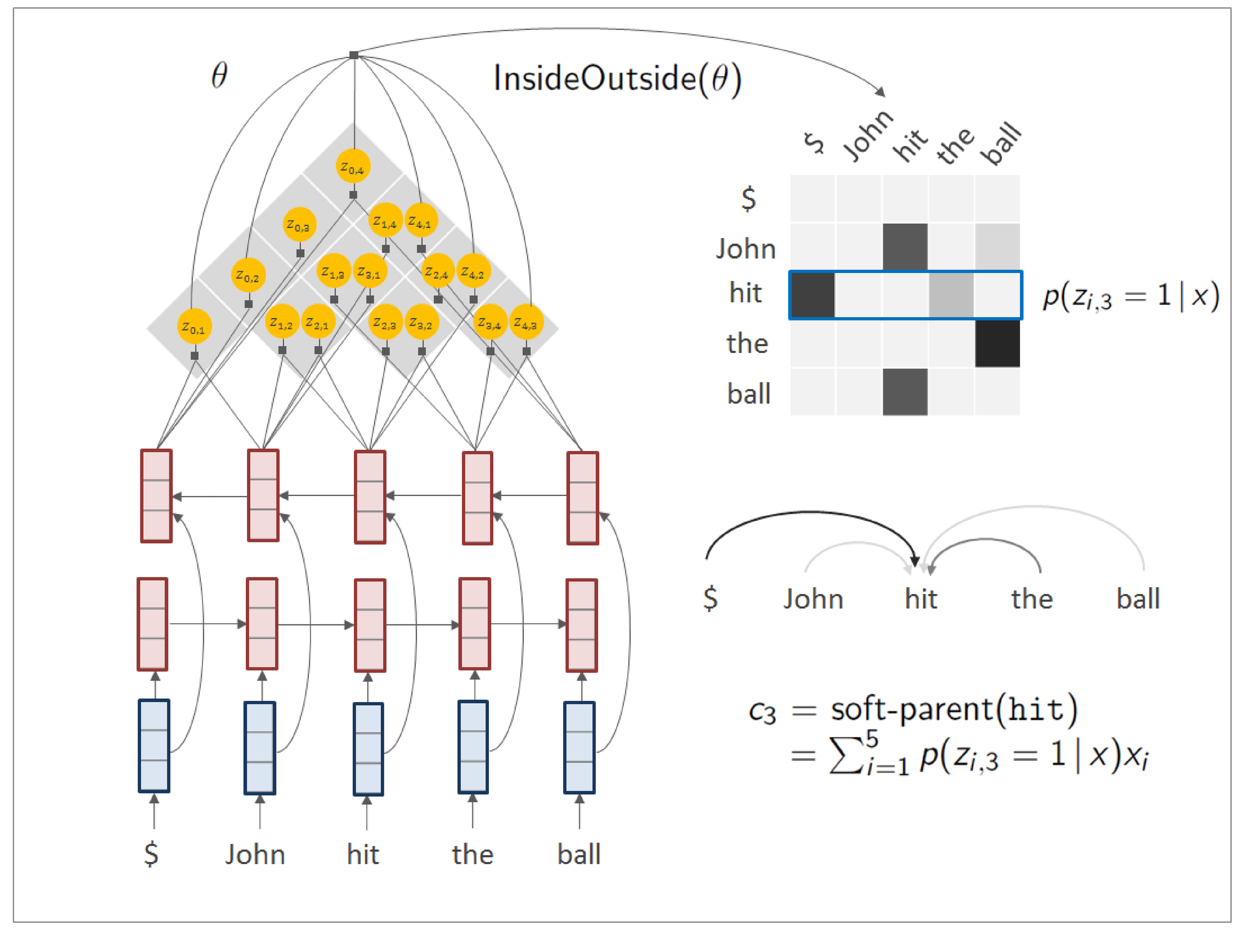
Syntactic Attention: Natural Language Inference

- Input structure: graph-based dependency parser with latent variables $z_{ij}, \forall i \neq j$ ($z_{ij} = 1 \implies x_i$ is head of x_j)
- Potentials: $\theta_{ij} = \mathrm{MLP}([h_i; h_j])$, $h = \mathrm{BiLSTM}(x)$
- Annotation function (parent word): $f_j(x, z) = \sum_{i=1}^T \mathbb{1}\{z_{ij} = 1\} x_i$
- Run inside-outside version of Eisner's algorithm to get marginals:

$$p(z_{ij} = 1 \mid x) = \text{INSIDEOUTSIDE}(\theta)$$

- Use parsing marginals to obtain context vector ("soft-parent"):

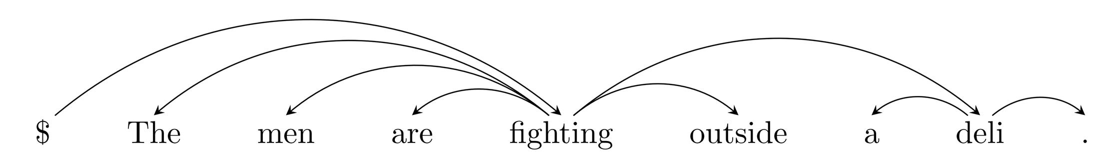
$$c_j = \mathbb{E}_z[f_j(x, z)] = \sum_{i=1}^T p(z_{ij} = 1 \mid x) x_i$$



- Baseline: only word embeddings (i.e. x_j)
- Hard parent: word + parent from pipelined dependency parser
- Simple: word + soft-match $p(z_{ij} = 1 \mid x) = \mathrm{softmax}(\theta_{ij})$
- Structured: word + soft-parent

Model	Acc %
Baseline	85.8
Hard parent	86.1
Simple	86.2
Structured	86.8

Run Viterbi on structured attention layer to get MAP parse:



Conclusions and Future Work

- Structured attention networks generalize simple attention and learn interesting internal representations.
- More experiments/models in the paper (tree transduction, question answering).
- Future work:
  - Approximate inference (e.g. loopy belief propagation) for richer graphical models.
  - Differentiable optimization as a neural network layer.

Code: `https://github.com/harvardnlp/struct-attn`

harvardnlp

{yoonkim@seas,carldenton@college,lhoang@g,srush@seas}.harvard.edu