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

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

Fixed-Size Encoder (MLP, RNN, CNN)

Encoder(input) $\in \mathbb{R}^D$

Decoder

Decoder(Encoder(input))

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

$\Downarrow$

Fixed-Size Encoder (MLP, RNN, CNN)

Encoder(input) $\in \mathbb{R}^D$

$\Downarrow$

Decoder

Decoder(Encoder(input))

Pure Encoder-Decoder Network

Input (sentence, image, etc.)

$\Downarrow$

Fixed-Size Encoder (MLP, RNN, CNN)

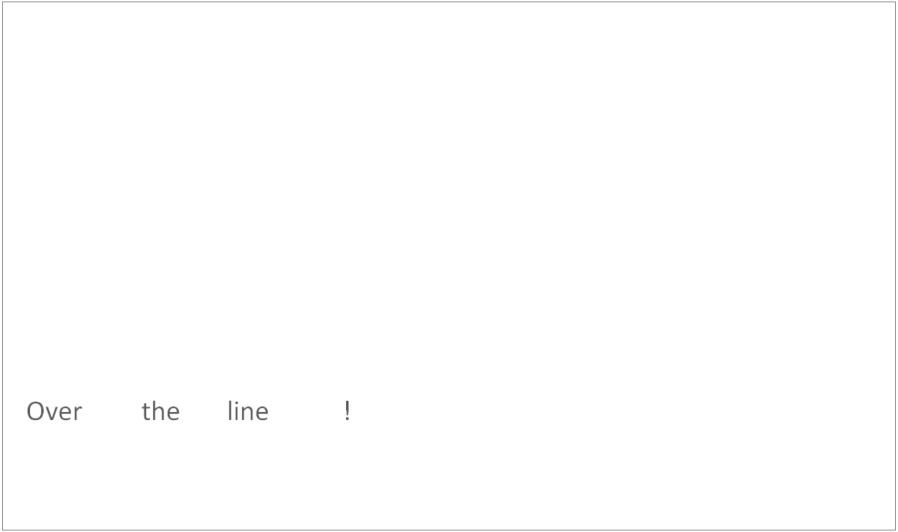
$\text{Encoder}(\text{input}) \in \mathbb{R}^D$

$\Downarrow$

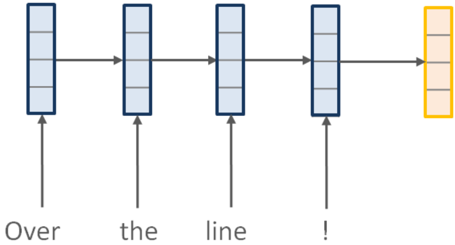
Decoder

$\text{Decoder}(\text{Encoder}(\text{input}))$

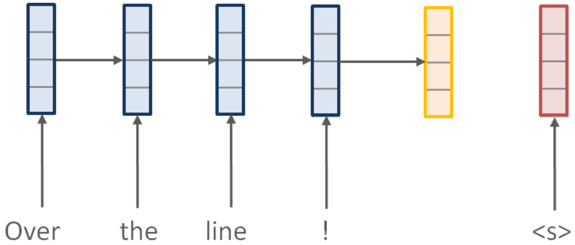
Example: Neural Machine Translation (Sutskever et al., 2014)



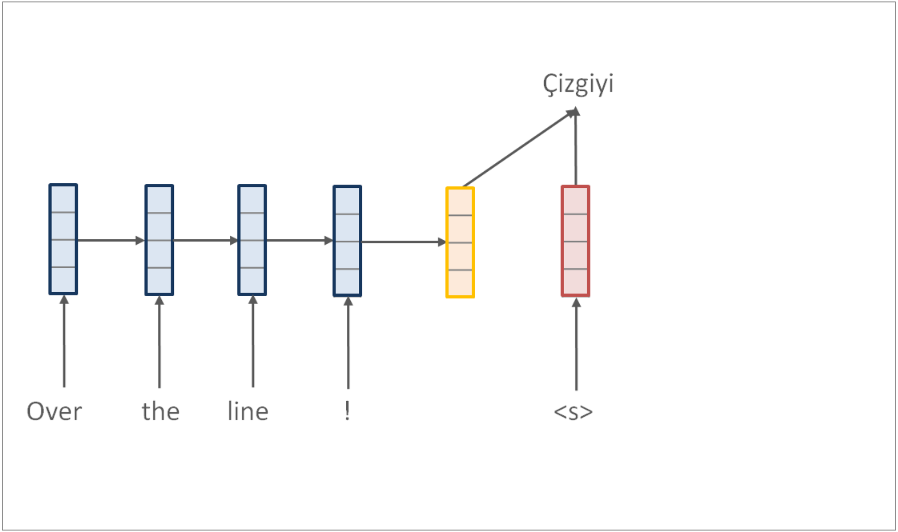
Example: Neural Machine Translation (Sutskever et al., 2014)



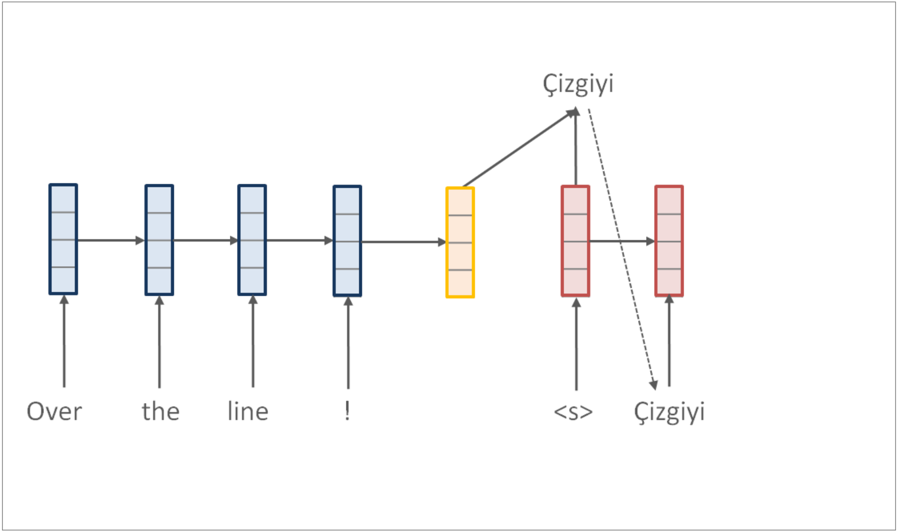
Example: Neural Machine Translation (Sutskever et al., 2014)



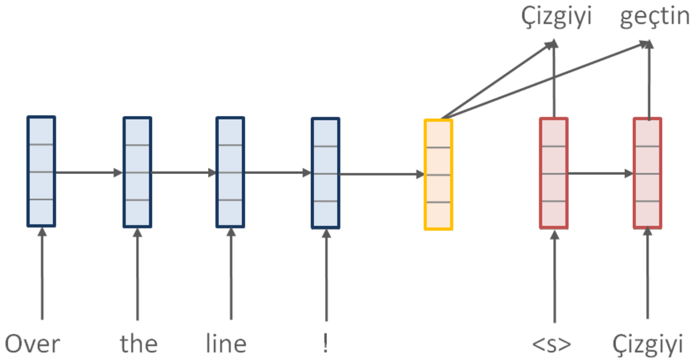
Example: Neural Machine Translation (Sutskever et al., 2014)



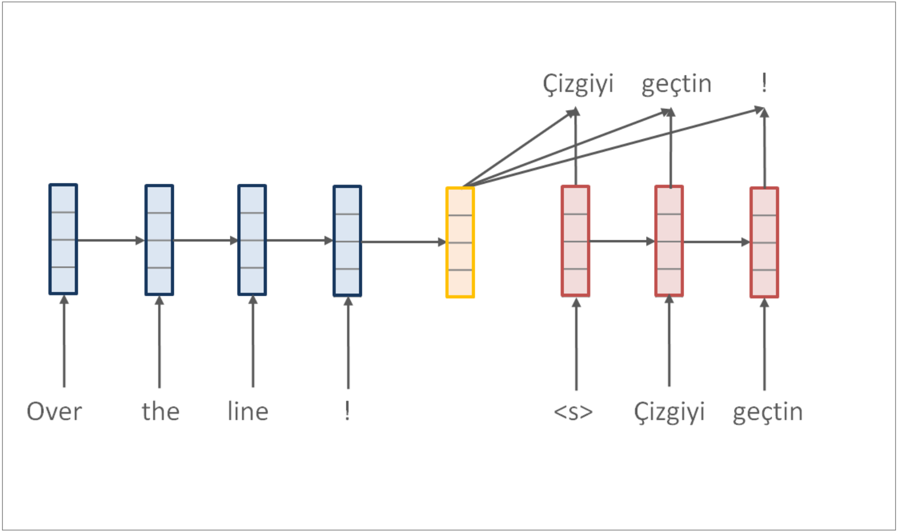
Example: Neural Machine Translation (Sutskever et al., 2014)



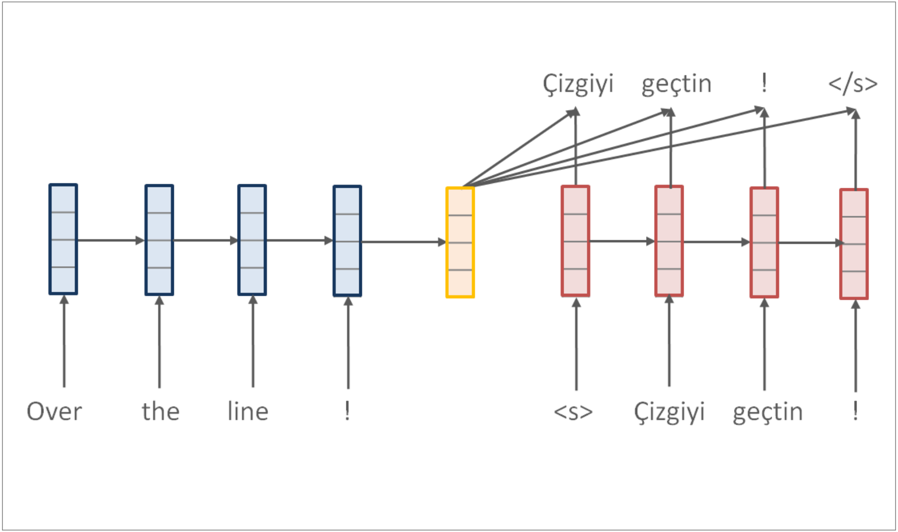
Example: Neural Machine Translation (Sutskever et al., 2014)



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.

$$\text{Encoder}(\text{input})$$

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

Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

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

Attention Distribution

"where"

Annotation Function

"what"

Context Vector ("soft selection")

Decoder

Neural Attention

Input (sentence, image, etc.)

⇩

Memory-Bank Encoder (MLP, RNN, CNN)

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

⇩

Attention Distribution "where"

Annotation Function "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 "where"

Annotation Function "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 — "where"

Annotation Function — "what"

Context Vector ("soft selection")

Decoder

Neural Attention

Input (sentence, image, etc.)

Memory-Bank Encoder (MLP, RNN, CNN)

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

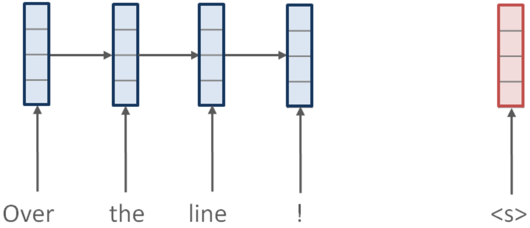
Attention Distribution "where"

Annotation Function "what"

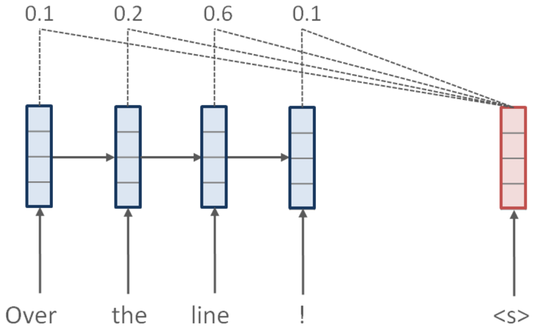
Context Vector ("soft selection")

Decoder

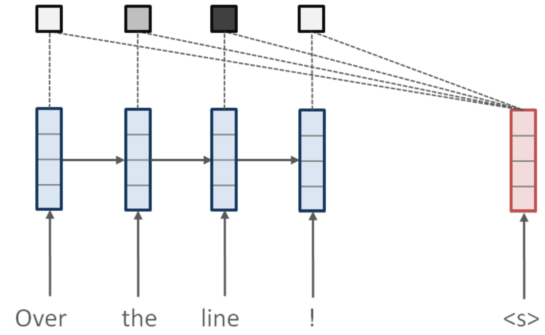
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



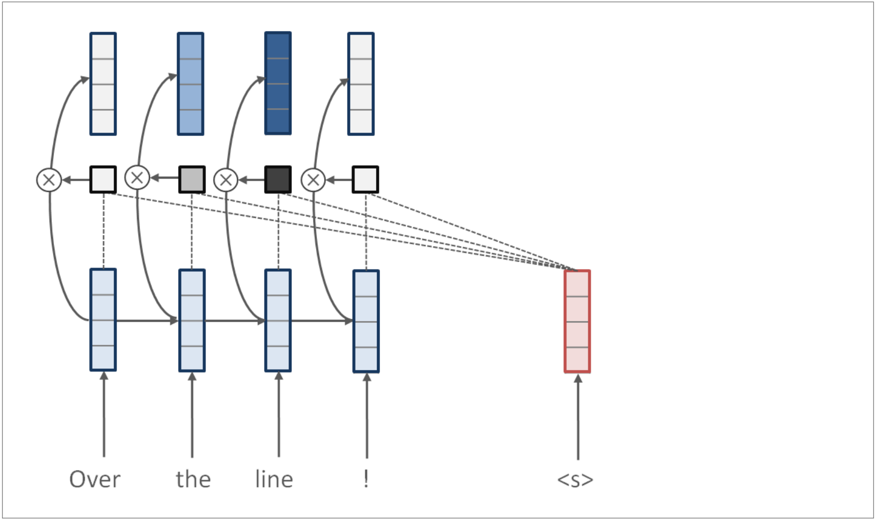
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



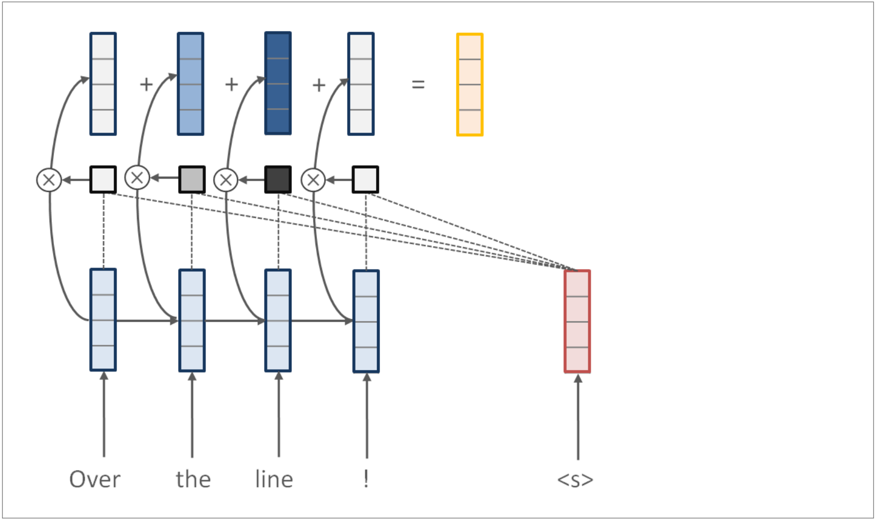
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



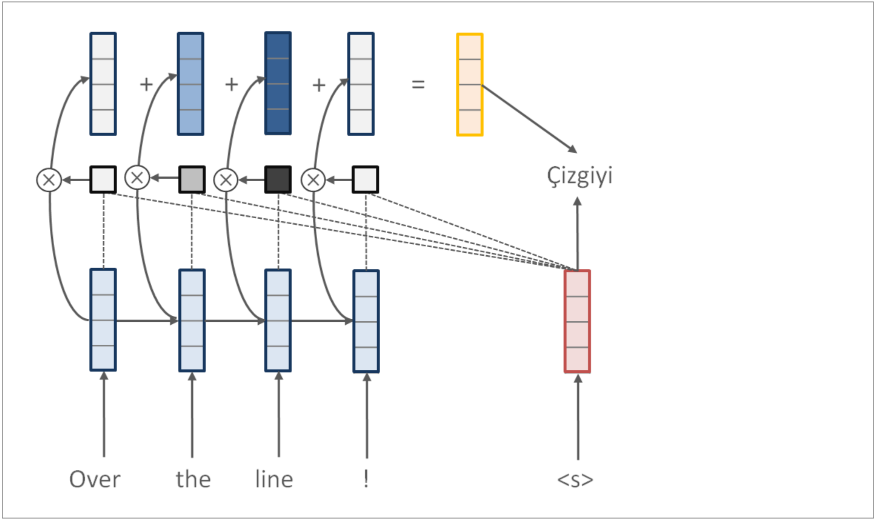
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



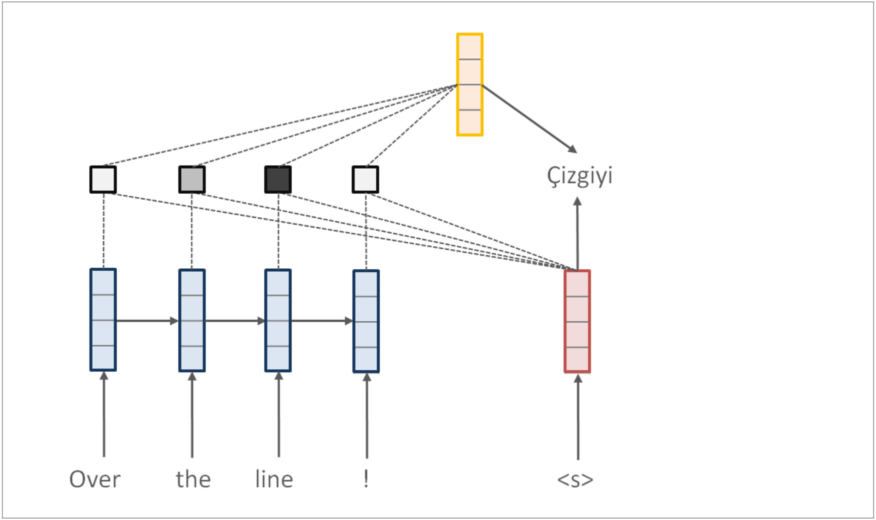
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



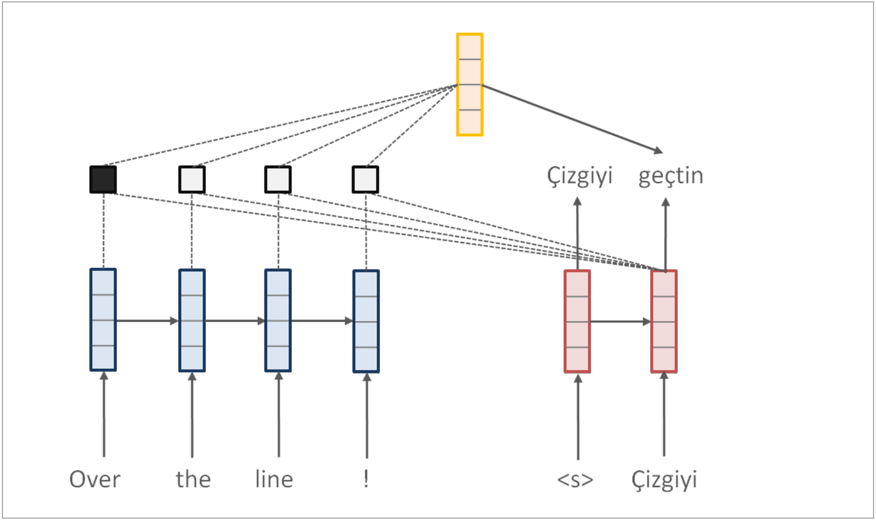
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



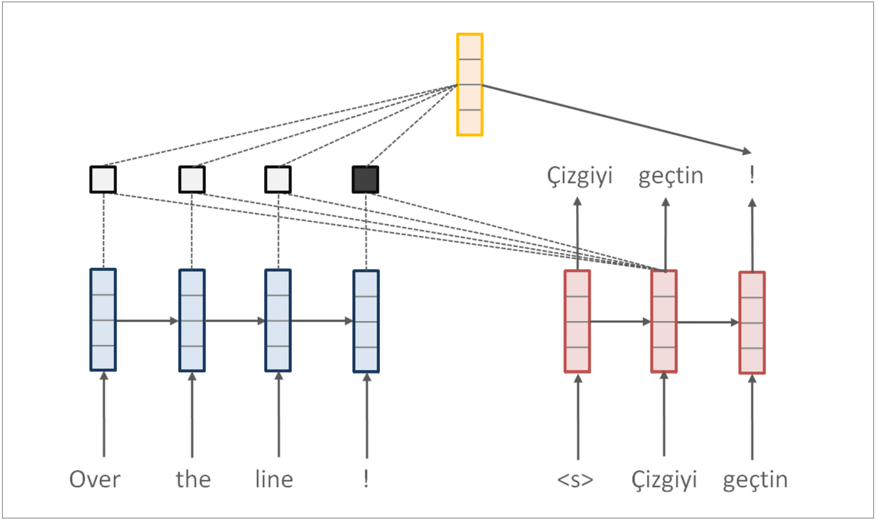
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



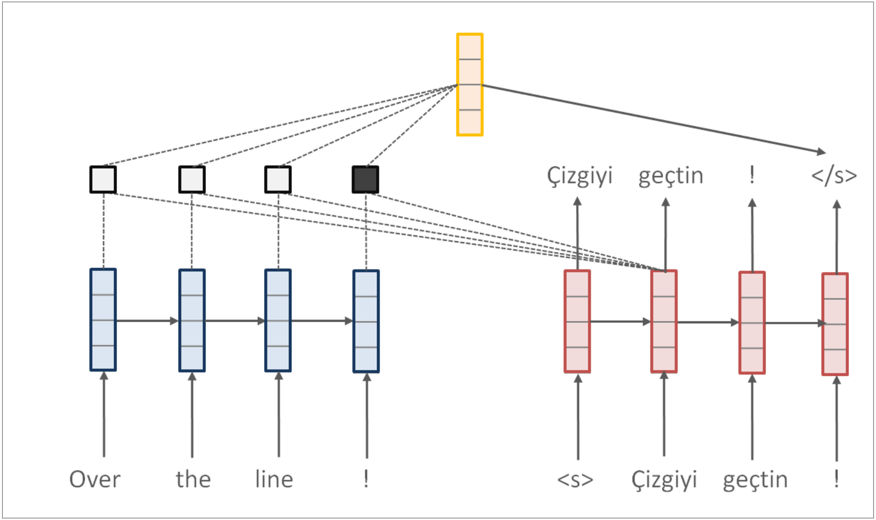
Attention-based Neural Machine Translation (Bahdanau et al., 2015)



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

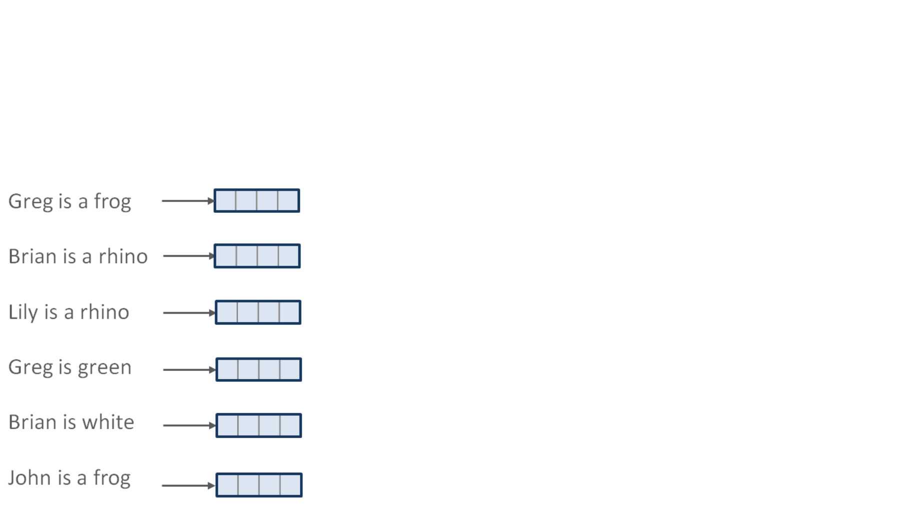
Lily is a rhino

Greg is green

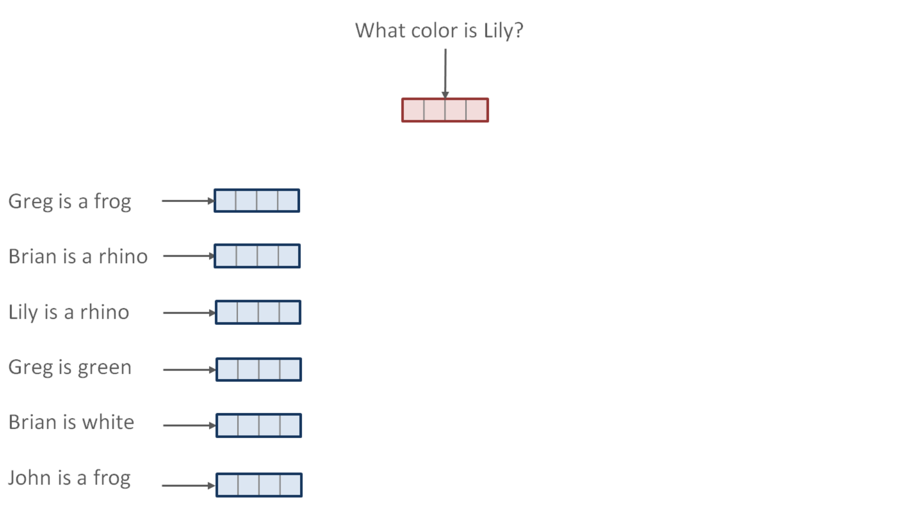
Brian is white

John is a frog

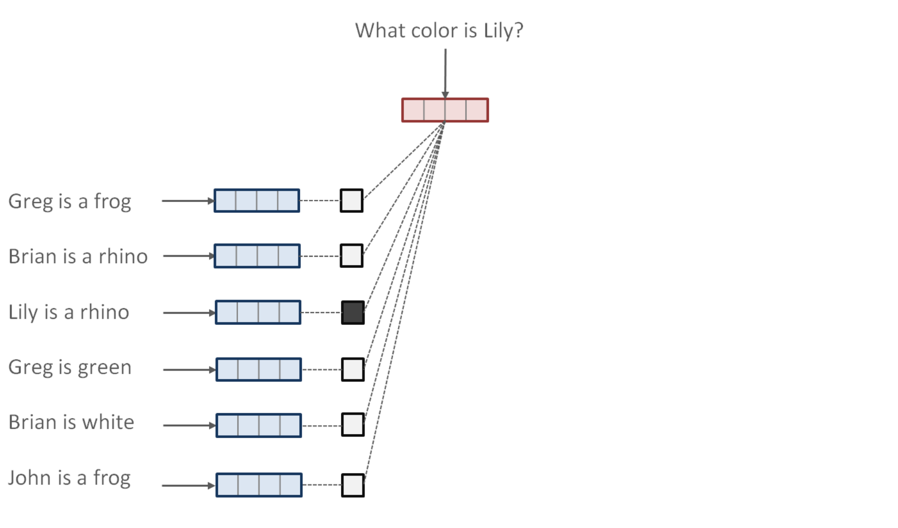
Question Answering (Sukhbaatar et al., 2015)



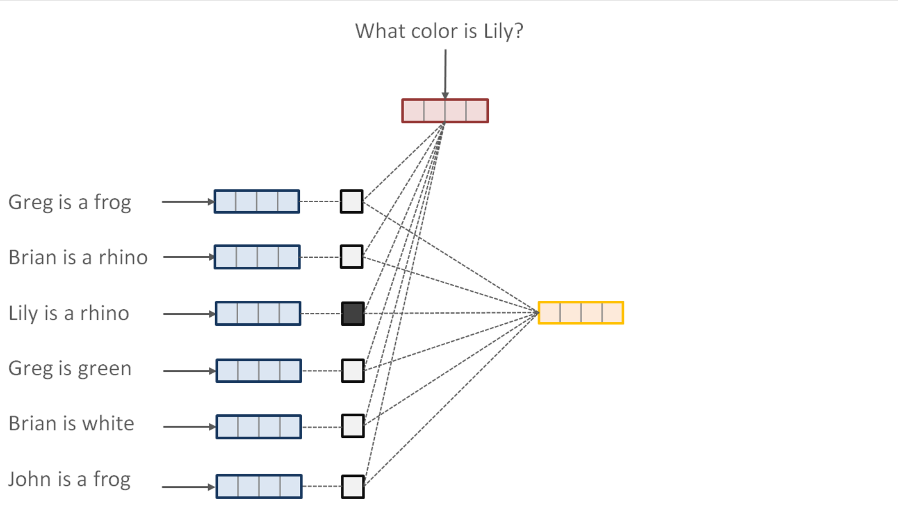
Question Answering (Sukhbaatar et al., 2015)



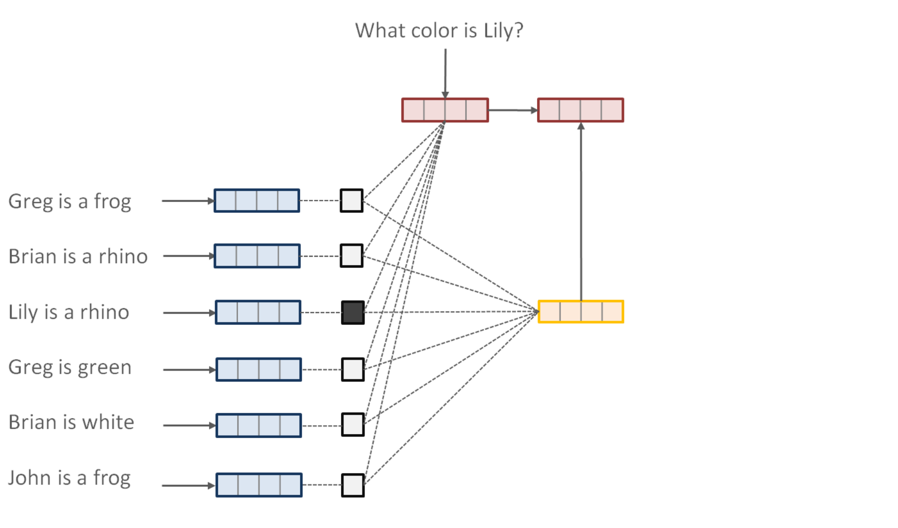
Question Answering (Sukhbaatar et al., 2015)



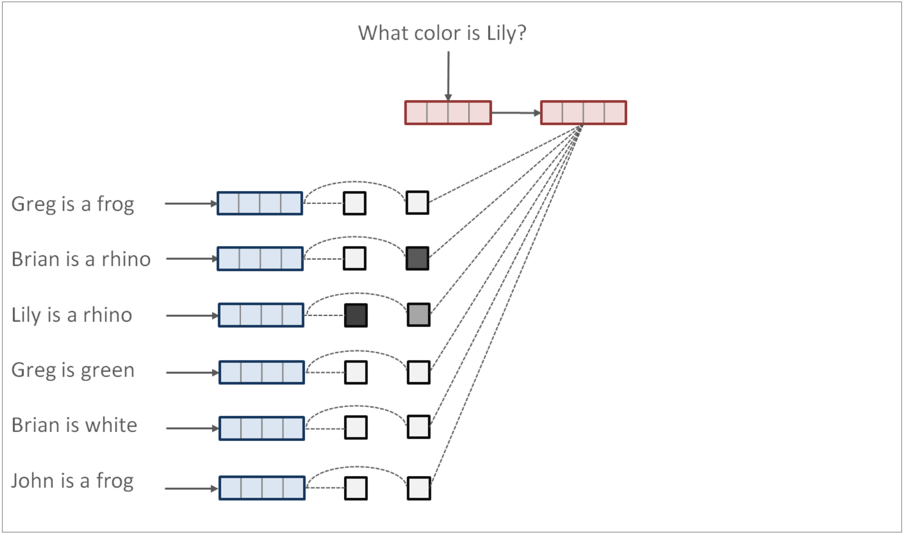
Question Answering (Sukhbaatar et al., 2015)



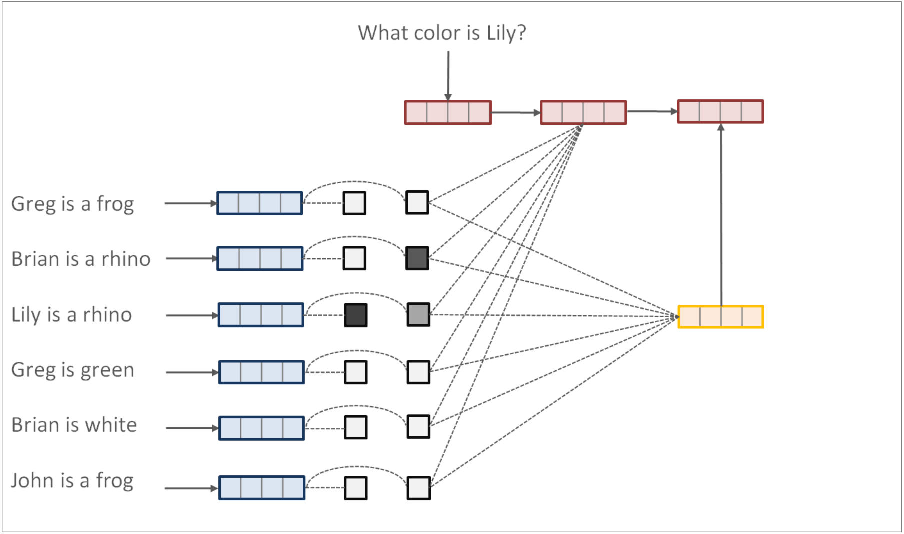
Question Answering (Sukhbaatar et al., 2015)



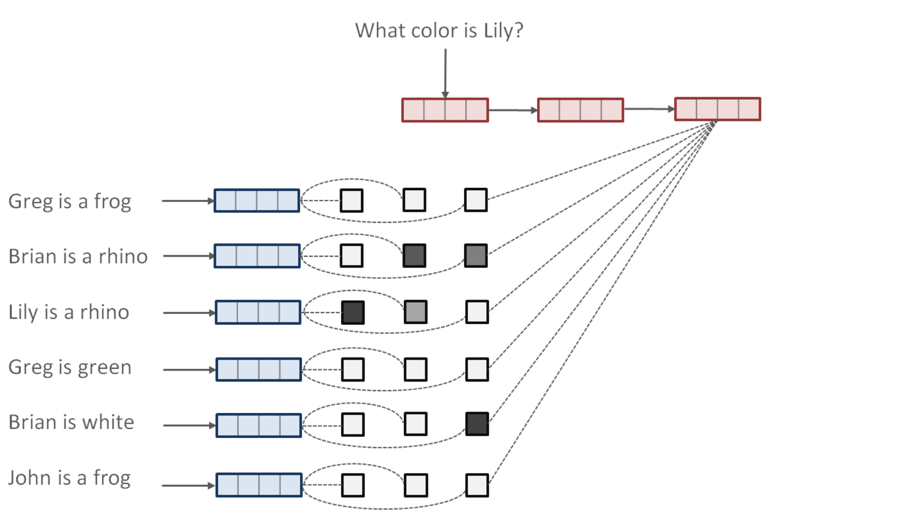
Question Answering (Sukhbaatar et al., 2015)



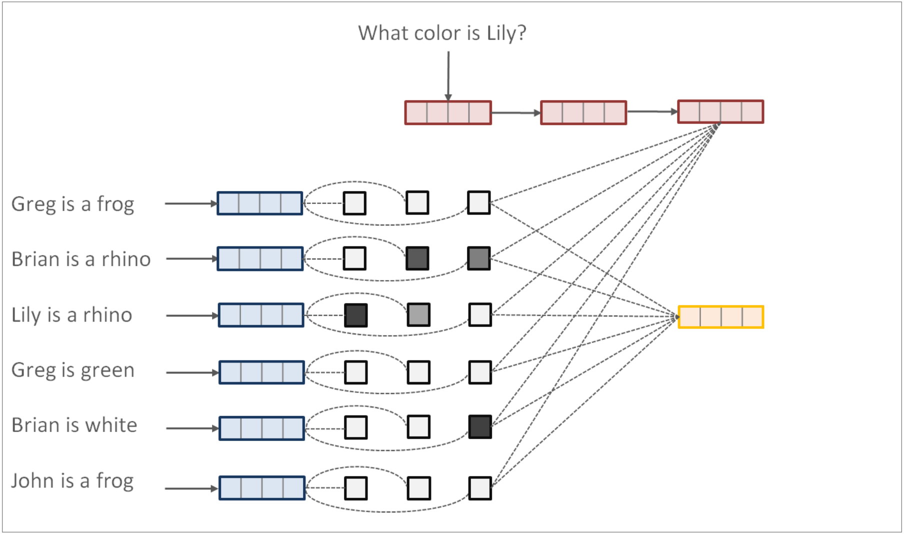
Question Answering (Sukhbaatar et al., 2015)



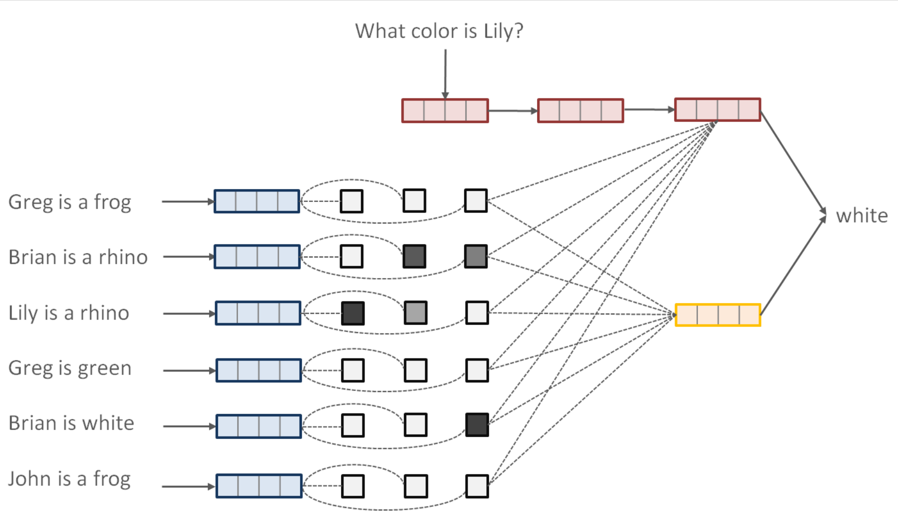
Question Answering (Sukhbaatar et al., 2015)



Question Answering (Sukhbaatar et al., 2015)



Question Answering (Sukhbaatar et al., 2015)



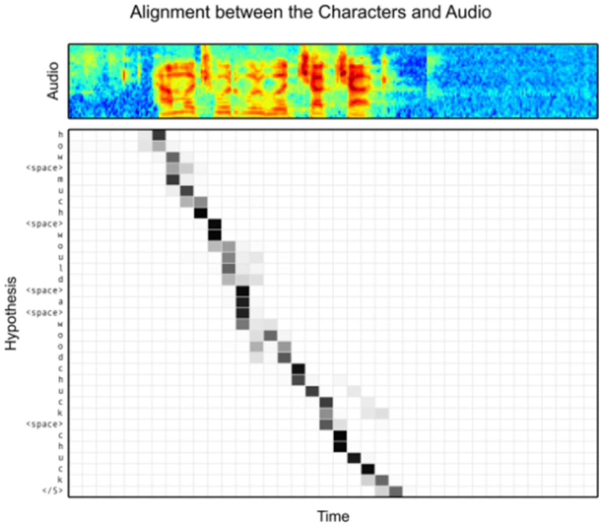
Other Applications of Attention Networks

- 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...

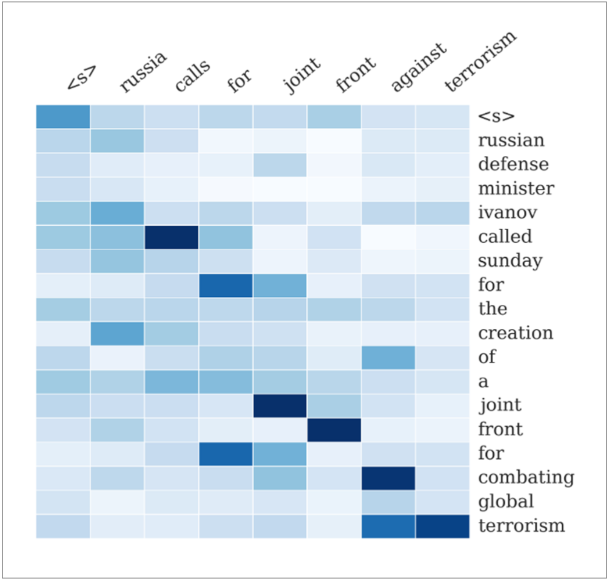
Other Applications: Image Captioning (Xu et al., 2015)

(b) A woman is throwing a frisbee in a park.

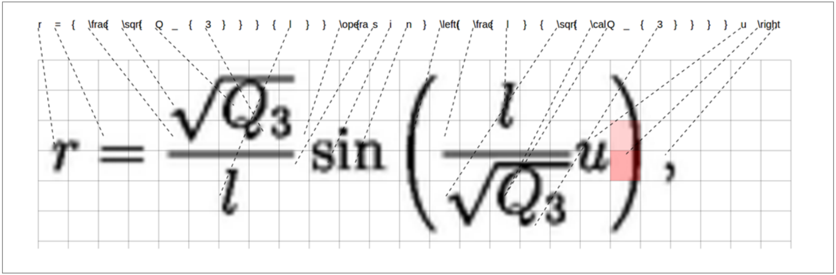
Other Applications: Speech Recognition (Chan et al., 2015)



Applications From HarvardNLP: Summarization (Rush et al., 2015)



Applications From HarvardNLP: Image-to-Latex (Deng et al., 2016)



Attention Networks: Notation

$x_1, \dots, x_T$	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 θ

Attention Networks: Notation

$x_1, \dots, x_T$	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 θ

Attention Networks: Machine Translation

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

End-to-End Requirements:

1. Need to compute attention $p(z = i \mid x, q; \theta)$
   $\Longrightarrow$ softmax function
2. Need to backpropagate to learn parameters θ
   $\Longrightarrow$ Backprop through softmax function

Attention Networks: Machine Translation

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

End-to-End Requirements:

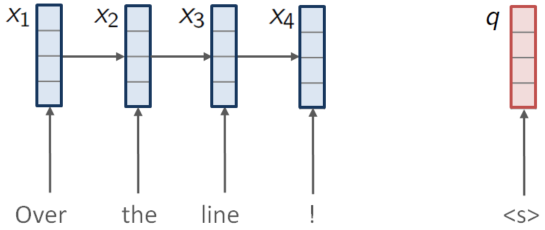
1. Need to compute attention $p(z = i \mid x, q; \theta)$

   $\implies$ softmax function

2. Need to backpropagate to learn parameters θ

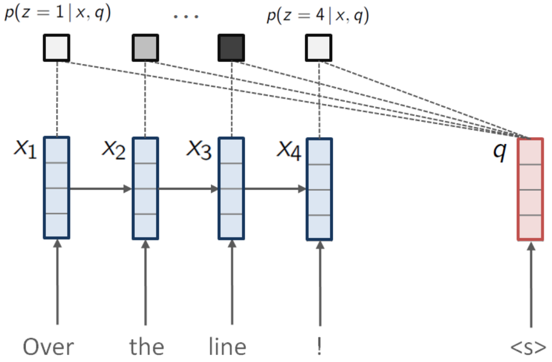
   $\implies$ Backprop through softmax function

Attention Networks: Machine Translation



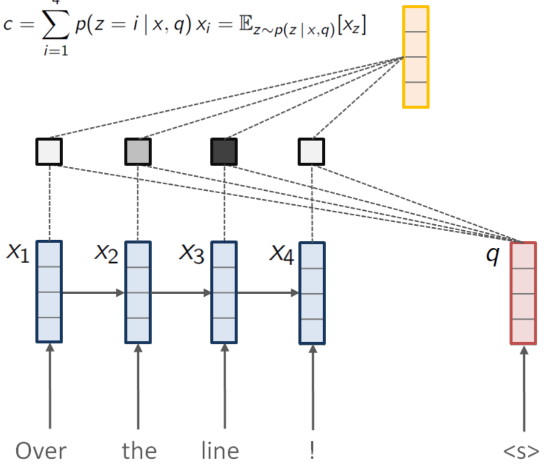
Attention Networks: Machine Translation

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

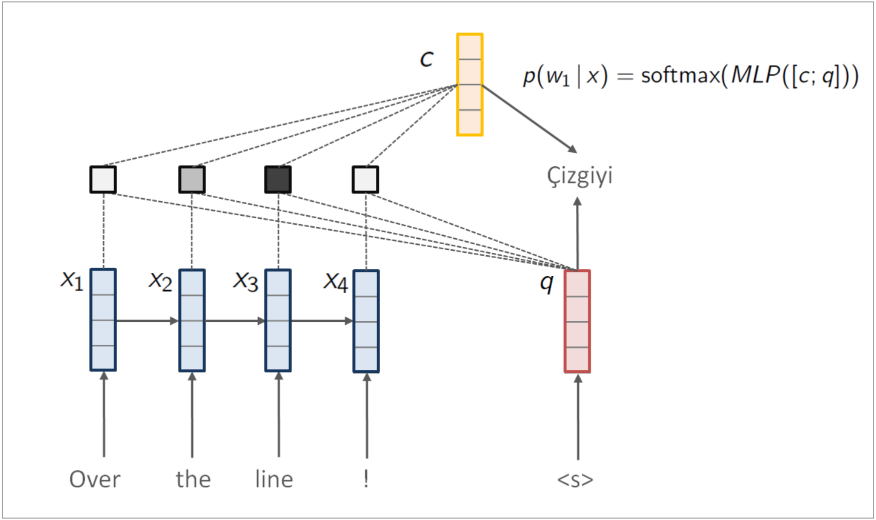


Attention Networks: Machine Translation

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



Attention Networks: Machine Translation



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

$p(z \,|\, x, q; \theta)$ Attention distribution over structures z

Structured Attention Networks: Notation

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

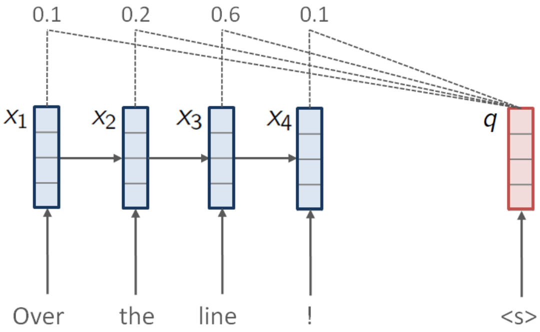
Structured Attention Networks: Notation

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

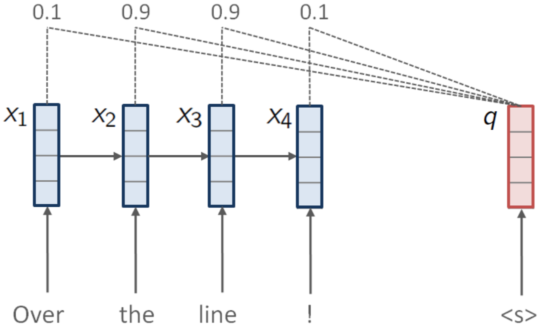
Consider family of functions $f(x, z)$ that makes $\mathbb{E}_{z \sim p(z \mid x,q)}[f(x, z)]$ computationally tractable

Structured Attention Networks for Neural Machine Translation

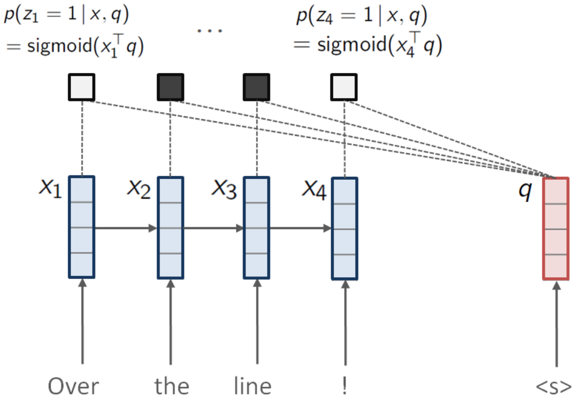
$\sum_{i=1}^{4} p(z = i \mid x, q) = 1$



Structured Attention Networks for Neural Machine Translation

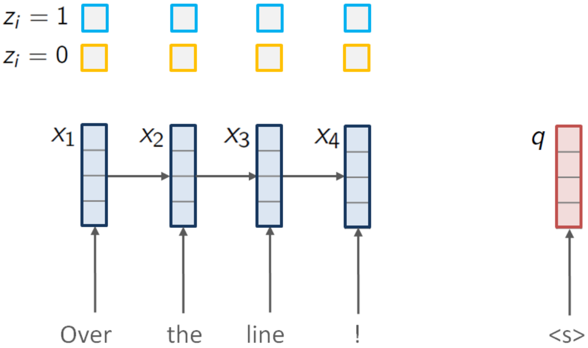


Structured Attention Networks for Neural Machine Translation



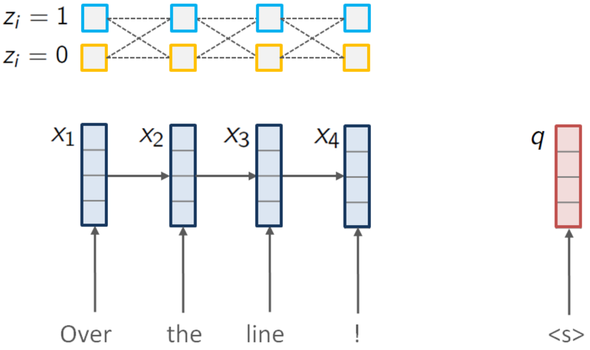
Structured Attention Networks for Neural Machine Translation

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



Structured Attention Networks for Neural Machine Translation

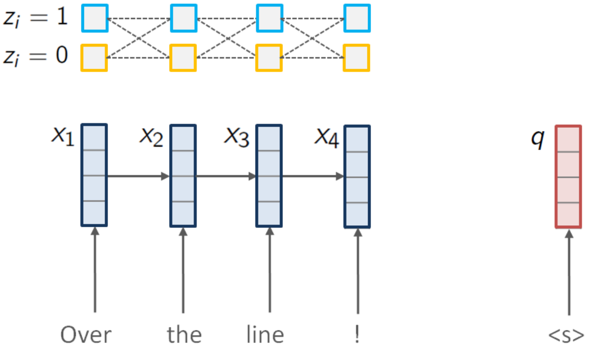
$$p(z_1, z_2, z_3, z_4 \mid x, q) = \text{softmax}\{\theta(z_1, z_2, z_3, z_4)\}$$
$$= \frac{1}{Z} \exp(\theta(z_1, z_2, z_3, z_4))$$

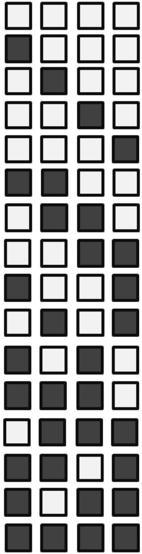


Structured Attention Networks for Neural Machine Translation

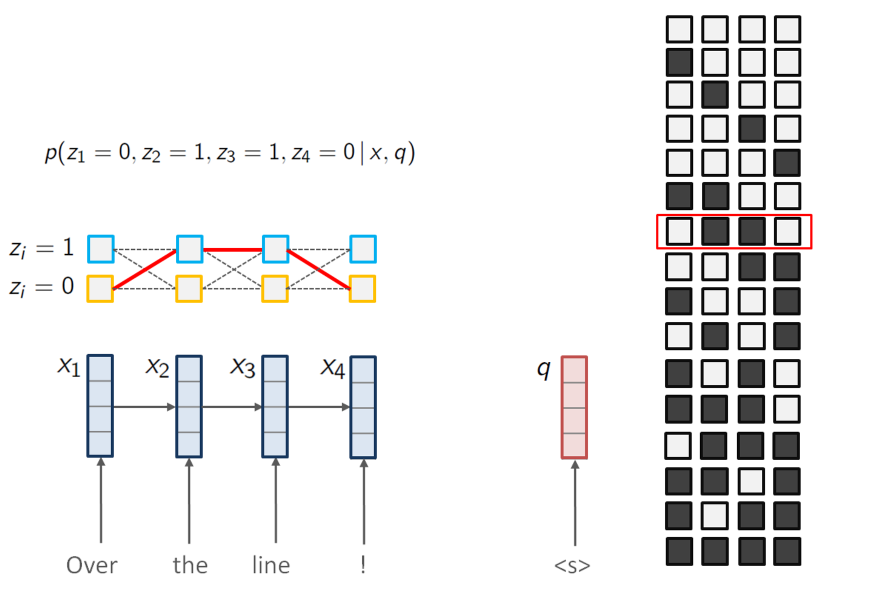
$$p(z_1, z_2, z_3, z_4 \mid x, q) = \text{softmax}\{\theta(z_1, z_2, z_3, z_4)\}$$
$$= \frac{1}{Z} \exp(\theta(z_1, z_2, z_3, z_4))$$

$$Z = \sum_{[z'_1, z'_2, z'_3, z'_4] \in \{0,1\}^4} \exp(\theta(z'_1, z'_2, z'_3, z'_4))$$



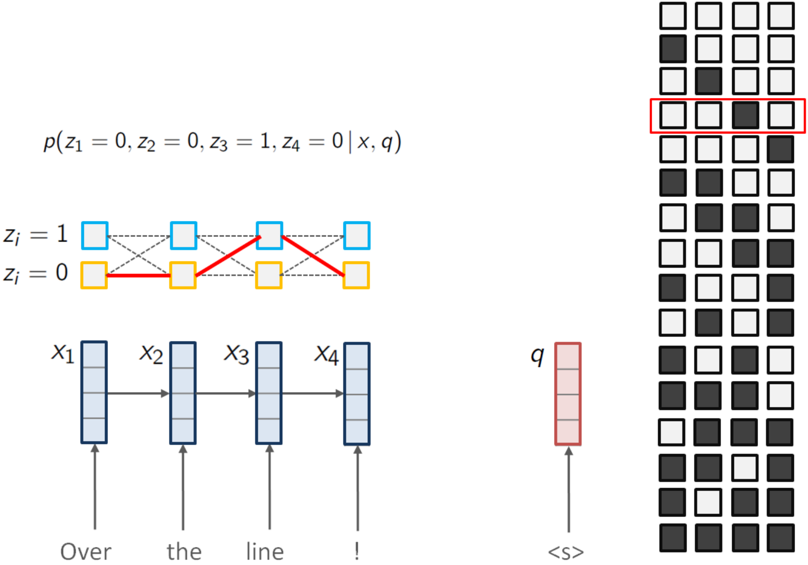


Structured Attention Networks for Neural Machine Translation



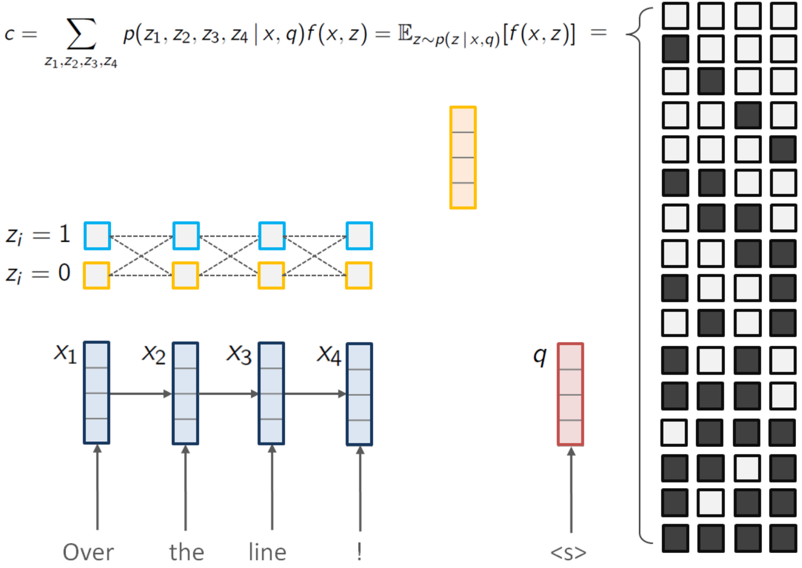
Structured Attention Networks for Neural Machine Translation

$$p(z_1 = 0, z_2 = 0, z_3 = 1, z_4 = 0 \,|\, x, q)$$



Structured Attention Networks for Neural Machine Translation

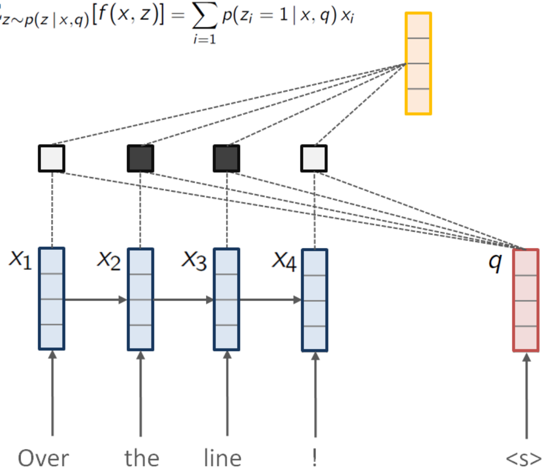
$$c = \sum_{z_1, z_2, z_3, z_4} p(z_1, z_2, z_3, z_4 \,|\, x, q) f(x, z) = \mathbb{E}_{z \sim p(z \,|\, x, q)}[f(x, z)] =$$



Structured Attention Networks for Neural Machine Translation

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

$$\mathbb{E}_{z \sim p(z \mid x, q)}[f(x, z)] = \sum_{i=1}^{4} p(z_i = 1 \mid x, q)\, x_i$$



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 $\theta_{i,i+1}(z_i, z_{i+1}; x)$

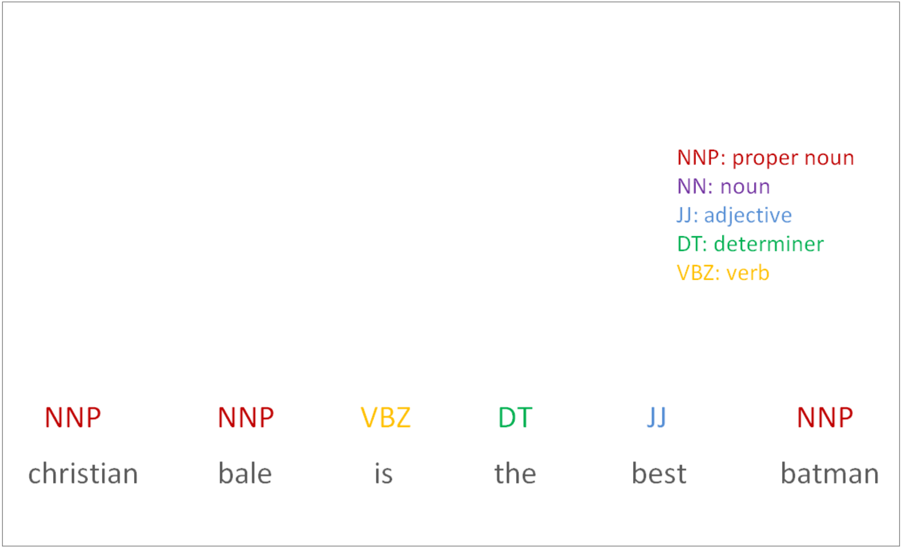
$$p(z_1 \ldots, z_T \mid x; \theta) = \text{softmax}\left(\sum_{i=1}^{T-1} \theta_{i,i+1}(z_i, z_{i+1}; x)\right)$$

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

$$Z = \sum_{z' \in \mathcal{C}} \exp\left(\sum_{i=1}^{T-1} \theta_{i,i+1}(z'_i, z'_{i+1}; x)\right)$$

Example: Part-of-Speech Tagging

Example: Part-of-Speech Tagging

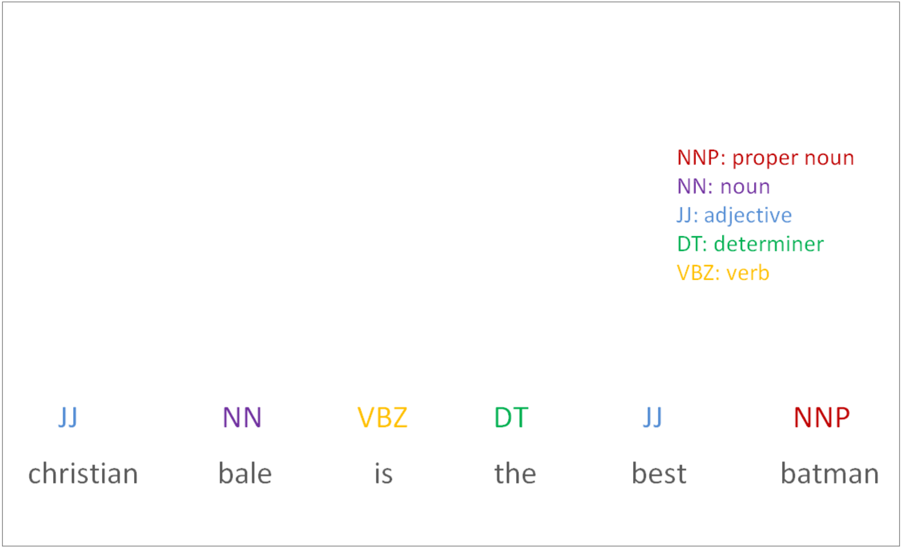


Example: Part-of-Speech Tagging

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

NNP	NNP	VBZ	DT	JJ	NNP
christian	bale	is	the	best	batman

Example: Part-of-Speech Tagging

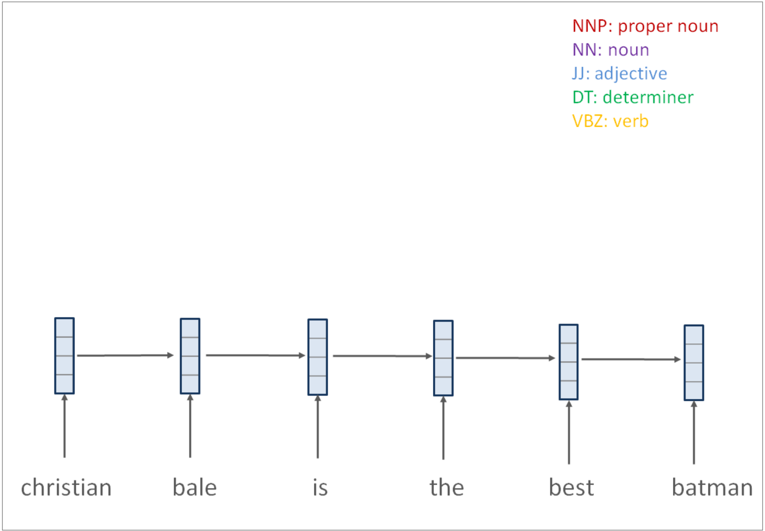


Example: Part-of-Speech Tagging

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

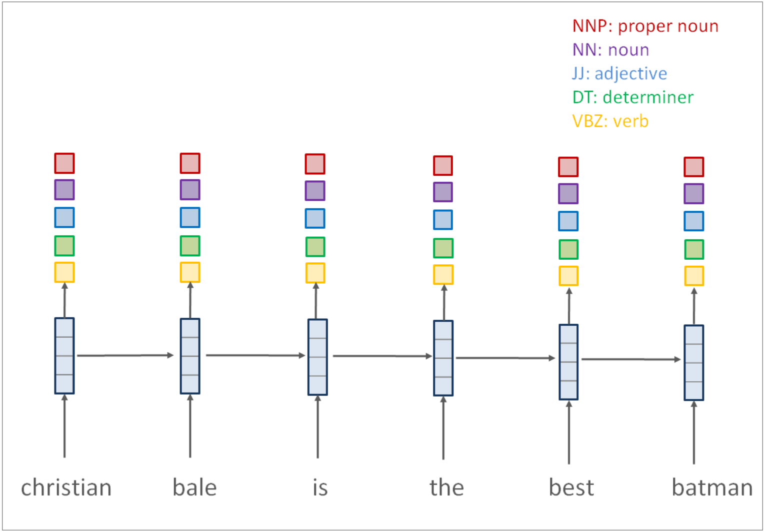
JJ	NN	VBZ	DT	JJ	NNP
christian	bale	is	the	best	batman

Neural CRF for Sequence Tagging (Collobert et al., 2011)



Neural CRF for Sequence Tagging (Collobert et al., 2011)

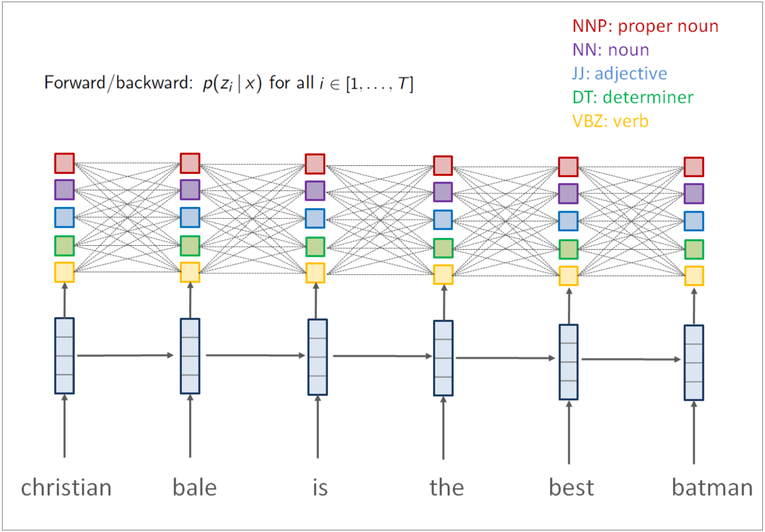
Unary potentials $\theta_i(c) = w_c^\top x_i$ come from neural network



Inference in Linear-Chain CRF

Pairwise potentials are simple parameters b, so altogether

$$\theta_{i,i+1}(c,d) = \theta_i(c) + \theta_{i+1}(d) + b_{c,d}$$



Structured Attention Networks: Notation

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

   $\Longrightarrow$ Forward-backward algorithm

2. Gradients wrt attention distribution parameters θ

   $\Longrightarrow$ Backpropagation through forward-backward algorithm

Challenge: End-to-End Training

Requirements:

1. Compute attention distribution (marginals) $p(z_i \mid x, q; \theta)$

   $\Longrightarrow$ Forward-backward algorithm

2. Gradients wrt attention distribution parameters θ

   $\Longrightarrow$ Backpropagation through forward-backward algorithm

Challenge: End-to-End Training

Requirements:

1. Compute attention distribution (marginals) $p(z_i \mid x, q; \theta)$

   $\implies$ Forward-backward algorithm

2. Gradients wrt attention distribution parameters θ

   $\implies$ Backpropagation through forward-backward algorithm

Review: Forward-Backward Algorithm

θ: input potentials (e.g. from NN)

α, β: 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**

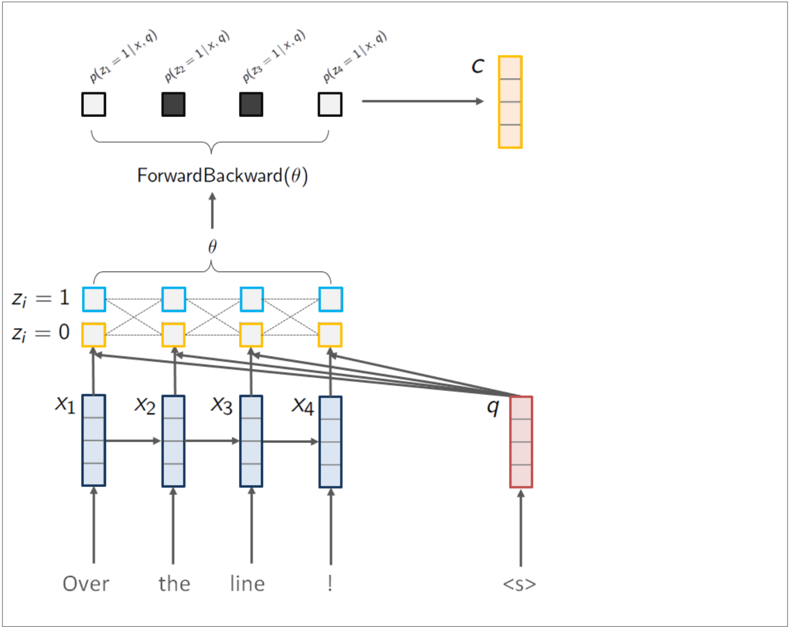
$\beta[i, z_i] \leftarrow \sum_{z_{i+1}1} \beta[i+1, z_{i+1}] \times \exp(\theta_{i,i+1}(z_i, z_{i+1}))$

Marginals

for $i = 1, \ldots, n; c \in \mathcal{C}$ **do**

$p(z_i = c \mid x) \leftarrow \alpha[i, c] \times \beta[i, c]/Z$

Structured Attention Networks for Neural Machine Translation



Forward-Backward Algorithm in Practice (Log-Space Semiring Trick)

$x \oplus y = \log(\exp(x) + \exp(y))$

$x \otimes y = x + y$

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

Forward

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

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

for $i = 1, \ldots, n; c \in \mathcal{C}$ **do**

$p(z_i = c \mid x) \leftarrow \exp(\alpha[i, c] \otimes \beta[i, c] \otimes -\log Z)$

Backpropagating through Forward-Backward

$\nabla_p^{\mathcal{L}}$: Gradient of arbitrary loss $\mathcal{L}$ with respect to marginals p

procedure $\textsc{BackpropForwardBackward}(\theta, p, \nabla_p^{\mathcal{L}})$

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**

$$\nabla_{\theta_{i-1,i}(z_i, z_{i+1})}^{\mathcal{L}} \leftarrow \exp(\hat{\alpha}[i, z_i] \otimes \beta[i+1, z_{i+1}] \oplus \alpha[i, z_i] \otimes \hat{\beta}[i+1, z_{i+1}] \oplus \ \alpha[i, z_i] \otimes \beta[i+1, z_{i+1}] \otimes -\log Z)$$

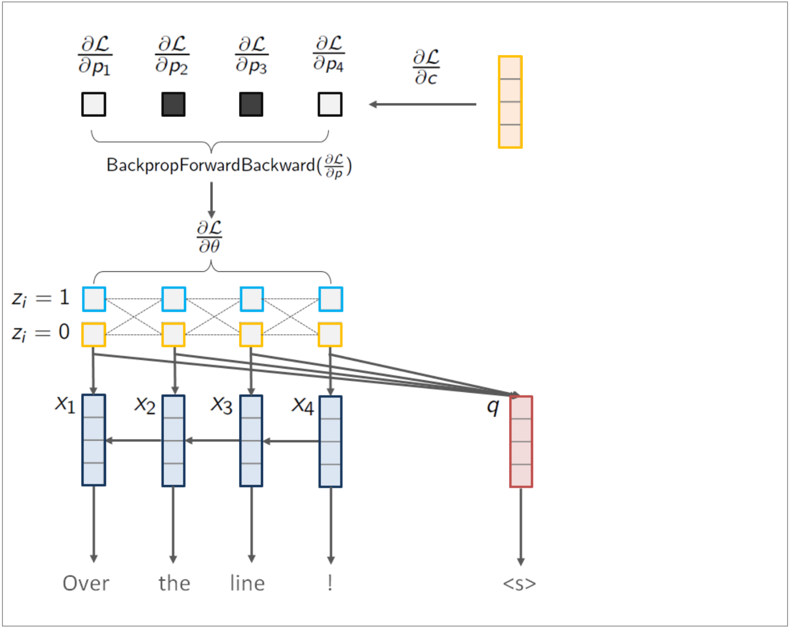
Interesting Issue: Negative Gradients Through Attention

- $\nabla_p^{\mathcal{L}}$: 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 $s_a = \mathrm{sign}(a)$

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

(Similar rules for $\otimes$)

Structured Attention Networks for Neural Machine Translation



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
- $p(z_1, \dots, z_T \mid x, q)$ 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 $\rightarrow$ English (from WAT 2015)
- Traditionally, word segmentation as a preprocessing step
- Use structured attention learn an implicit segmentation model

Experiments:

- Japanese characters $\rightarrow$ English words
- Japanese words $\rightarrow$ English words

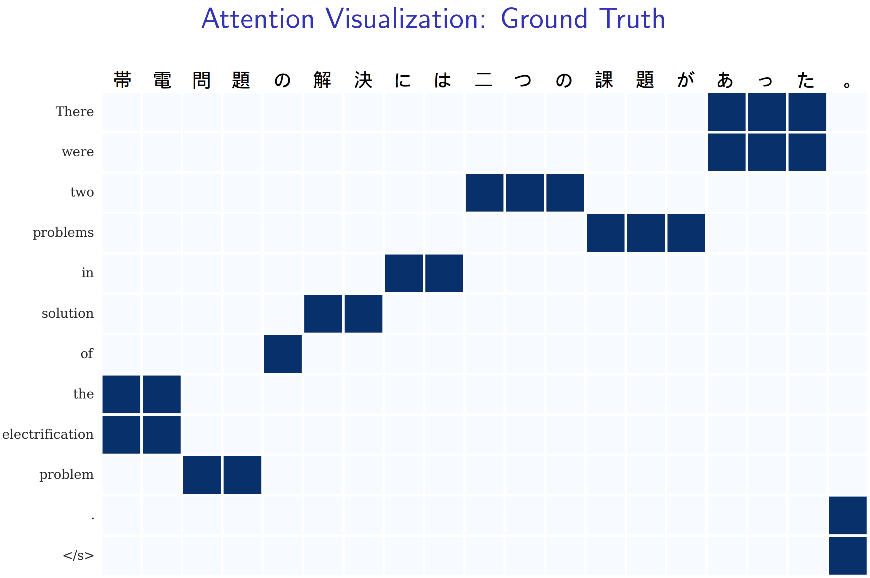
Segmental Attention for Neural Machine Translation

	Simple	Sigmoid	Structured
Char → Word	12.6	13.1	14.6
Word → Word	14.1	13.8	14.3

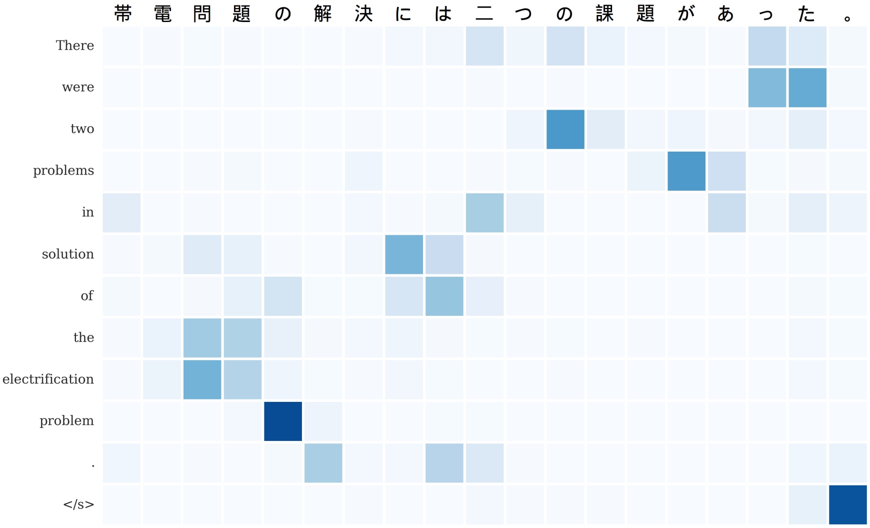
BLEU scores on test set (higher is better)

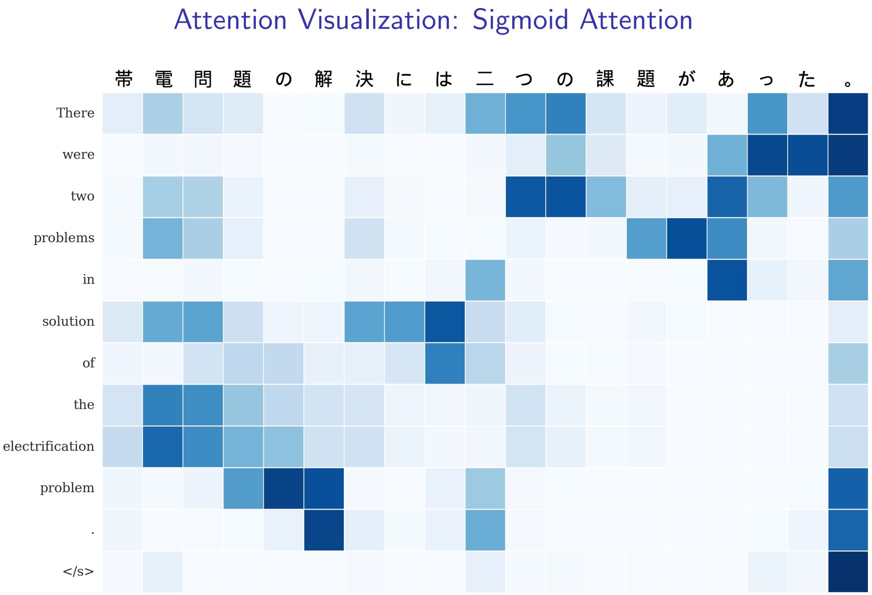
Models:

- Simple softmax attention: $\mathrm{softmax}(\theta_i)$
- Sigmoid attention: $\mathrm{sigmoid}(\theta_i)$
- Structured attention: $\mathrm{ForwardBackward}(\theta)$

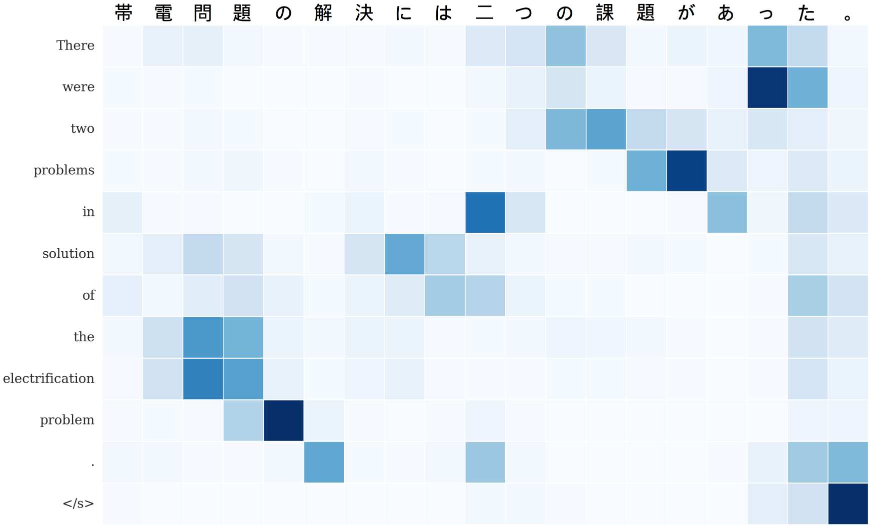

Attention Visualization: Ground Truth
帯 電 問 題 の 解 決 に は 二 つ の 課 題 が あ っ た 。
There
were
two
problems
in
solution
of
the
electrification
problem
.
</s>

Attention Visualization: Simple Attention



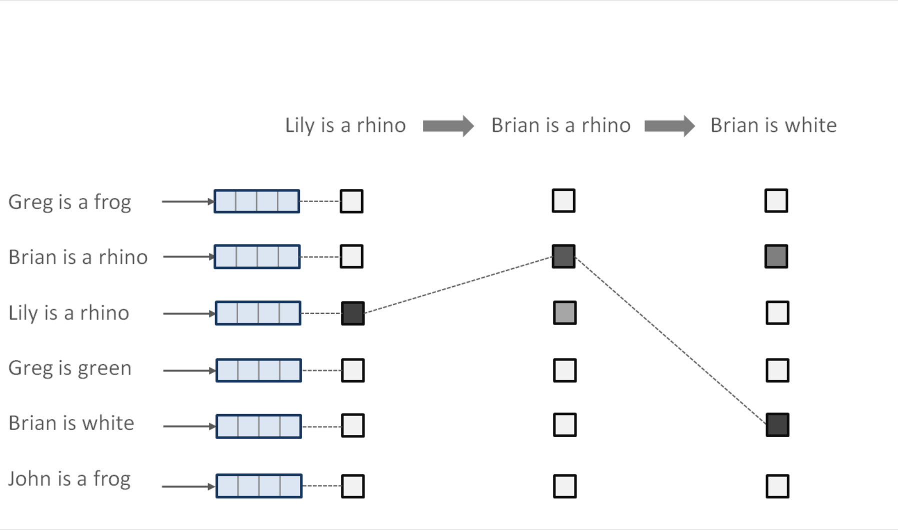

Attention Visualization: Sigmoid Attention
帯 電 問 題 の 解 決 に は 二 つ の 課 題 が あ っ た 。
There
were
two
problems
in
solution
of
the
electrification
problem
.
</s>

Attention Visualization: Structured Attention



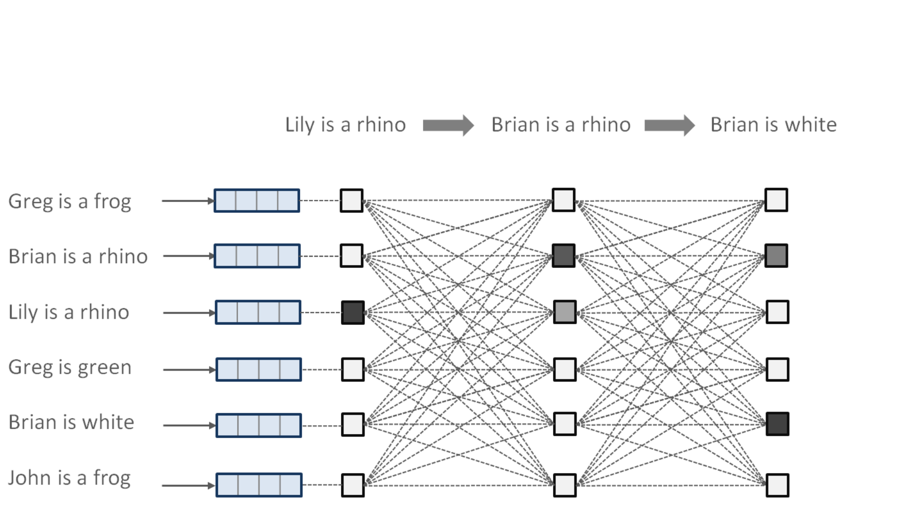
Sequential Attention over Facts for Question Answering

Simple attention: Greedy soft-selection of K supporting facts



Sequential Attention over Facts for Question Answering

Structured attention: Consider all possible sequences

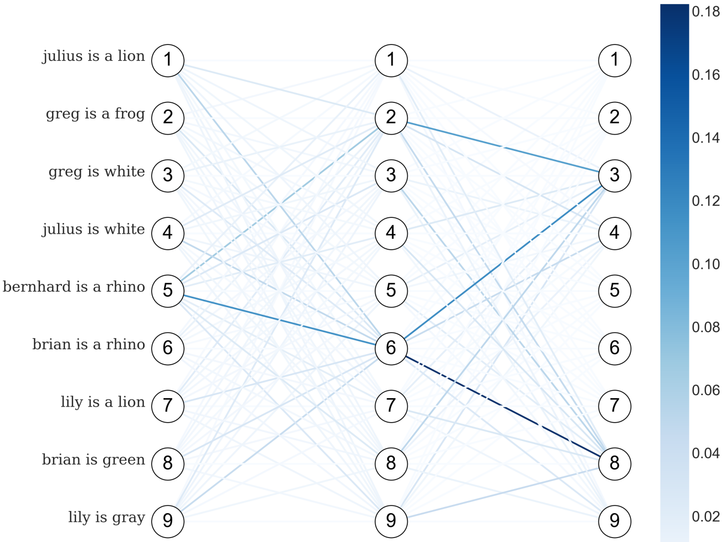


Sequential Attention over Facts for Question Answering

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

Task	K	Simple Ans %	Simple Fact %	Structured Ans %	Structured Fact %
Task 02	2	87.3	46.8	84.7	81.8
Task 03	3	52.6	1.4	40.5	0.1
Task 11	2	97.8	38.2	97.7	80.8
Task 13	2	95.6	14.8	97.0	36.4
Task 14	2	99.9	77.6	99.7	98.2
Task 15	2	100.0	59.3	100.0	89.5
Task 16	3	97.1	91.0	97.9	85.6
Task 17	2	61.1	23.9	60.6	49.6
Task 18	2	86.4	3.3	92.2	3.9
Task 19	2	21.3	10.2	24.4	11.5
Average	–	81.4	39.6	81.0	53.7

Sequential Attention over Facts for Question Answering



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)

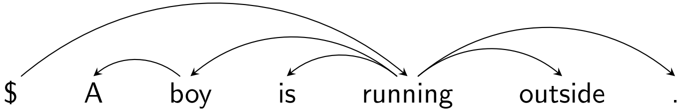
P	The boy is running through a grassy area.	
H	The boy is in his room.	C
	A boy is running outside.	E
	The boy is in a park.	N

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

Natural Language Inference

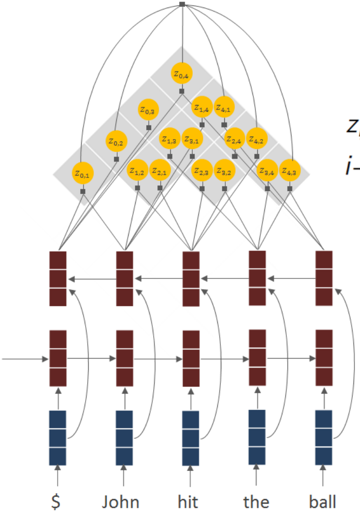
Given a premise (P) and a hypothesis (H), predict the relationship: Entailment (E), Contradiction (C), Neutral (N)

P	The boy is running through a grassy area.	
H	The boy is in his room.	C
	A boy is running outside.	E
	The boy is in a park.	N



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

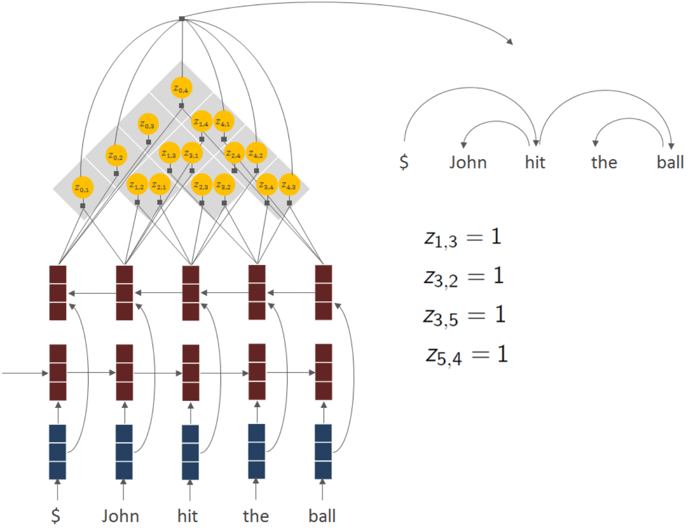
Neural CRF Parsing (Durrett and Klein, 2015; Kipperwasser and Goldberg, 2016)



$z_{ij} = 1 \implies$

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)

   $\Longrightarrow$ Inside/outside algorithm

2. Gradients wrt attention distribution parameters: $\frac{\partial \mathcal{L}}{\partial \theta}$

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

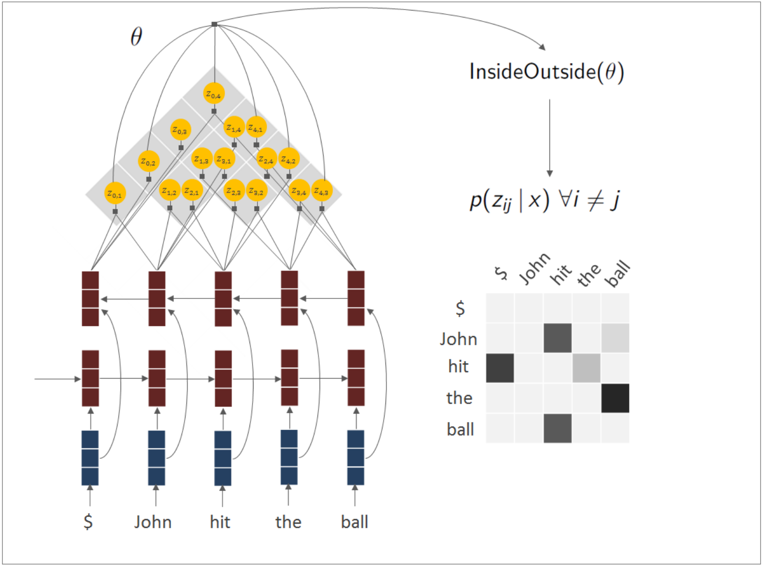
Forward/backward pass on inside-outside version of Eisner's algorithm (Eisner, 1996) takes $O(T^3)$ time.

Forward/Back-propagation through Inside-Outside Algorithm

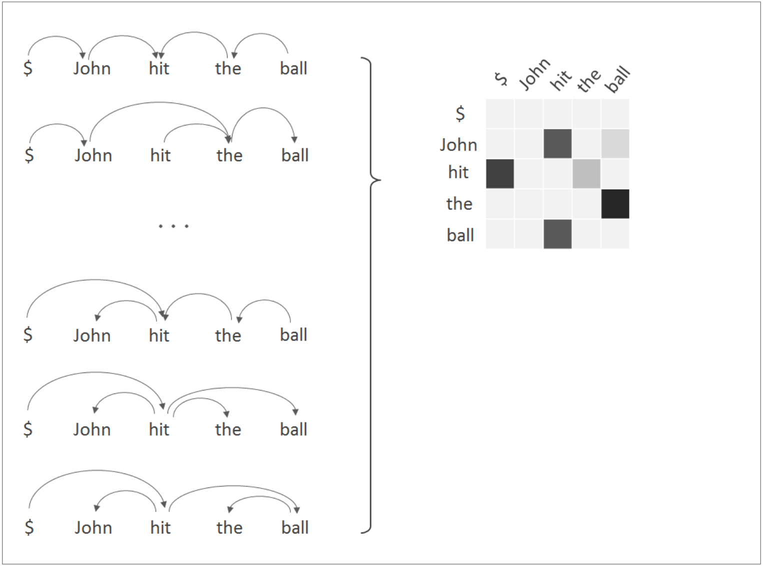
```
procedure InsideOutside(θ)
  α, β ← −∞                                          ▷ Initialize log of inside (α), outside (β) tables
  for i = 1, ..., n do
    α[i, i, L, 1] ← 0
    α[i, i, R, 1] ← 0
  β[1, n, R, 1] ← 0
  for k = 1, ..., n do                               ▷ Inside step
    for s = 1, ..., n − k do
      t ← s + k
      α[s, t, R, 0] ← ⊕_{u∈[s,t−1]} α[s, u, R, 1] ⊗ α[u + 1, t, L, 1] ⊗ θ_st
      α[s, t, L, 0] ← ⊕_{u∈[s,t−1]} α[s, u, R, 1] ⊗ α[u + 1, t, L, 1] ⊗ θ_ts
      α[s, t, R, 1] ← ⊕_{u∈[s+1,t]} α[s, u, R, 0] ⊗ α[u, t, R, 1]
      α[s, t, L, 1] ← ⊕_{u∈[s,t−1]} α[s, u, L, 1] ⊗ α[u, t, L, 0]
  for k = n, ..., 1 do                               ▷ Outside step
    for s = 1, ..., n − k do
      t ← s + k
      for u = s + 1, ..., t do
        β[s, u, R, 0] ←_⊕ β[s, t, R, 1] ⊗ α[u, t, R, 1]
        β[u, t, R, 1] ←_⊕ β[s, t, R, 1] ⊗ α[s, u, R, 0]
      if s > 1 then
        for u = s, ..., t − 1 do
          β[s, u, L, 1] ←_⊕ β[s, t, L, 1] ⊗ α[u, t, L, 0]
          β[u, t, L, 0] ←_⊕ β[s, t, L, 1] ⊗ α[s, u, L, 1]
      for u = s, ..., t − 1 do
        β[s, u, R, 1] ←_⊕ β[s, t, R, 0] ⊗ α[u + 1, t, L, 1] ⊗ θ_st
        β[u + 1, t, L, 1] ←_⊕ β[s, t, R, 0] ⊗ α[s, u, R, 1] ⊗ θ_st
      if s > 1 then
        for u = s, ..., t − 1 do
          β[s, u, R, 1] ←_⊕ β[s, t, L, 0] ⊗ α[u + 1, t, L, 1] ⊗ θ_ts
          β[u + 1, t, L, 1] ←_⊕ β[s, t, L, 0] ⊗ α[s, u, R, 1] ⊗ θ_ts
  A ← α[1, n, R, 1]                                  ▷ Log partition
  for s = 1, ..., n − 1 do                           ▷ Compute marginals. Note that p[s, t] = p(z_st = 1 | x)
    for t = s + 1, ..., n do
      p[s, t] ← exp(α[s, t, R, 0] ⊗ β[s, t, R, 0] ⊗ −A)
      if s > 1 then
        p[t, s] ← exp(α[s, t, L, 0] ⊗ β[s, t, L, 0] ⊗ −A)
  return p
```

```
procedure BackpropInsideOutside(θ, p, ∇^ℓ_p)
  for s, t = 1, ..., n; s ≠ t do                     ▷ Backpropagation uses the identity ∇^ℓ_θ = (p ⊙ ∇^ℓ_p)∇^{log p}_θ
    δ[s, t] ← log p[s, t] ⊗ log ∇^ℓ_p[s, t]          ▷ δ = log(p ⊙ ∇^ℓ_p)
  ∇^ℓ_α, ∇^ℓ_β, log ∇^ℓ_θ ← −∞                       ▷ Initialize inside (∇^ℓ_α), outside (∇^ℓ_β) gradients, and log of ∇^ℓ_θ
  for s = 1, ..., n − 1 do                           ▷ Backpropagate δ to ∇^ℓ_α and ∇^ℓ_β
    for t = s + 1, ..., n do
      ∇^ℓ_α[s, t, R, 0], ∇^ℓ_β[s, t, R, 0] ← δ[s, t]
      ∇^ℓ_α[1, n, R, 1] ←_⊕ −δ[s, t]
      if s > 1 then
        ∇^ℓ_α[s, t, L, 0], ∇^ℓ_β[s, t, L, 0] ← δ[t, s]
        ∇^ℓ_α[1, n, R, 1] ←_⊕ −δ[s, t]
  for k = 1, ..., n do                               ▷ Backpropagate through outside step
    for s = 1, ..., n − k do
      t ← s + k
      ν ← ∇^ℓ_β[s, t, R, 0] ⊗ β[s, t, R, 0]           ▷ ν, γ are temporary values
      for u = t, ..., n do
        ∇^ℓ_β[s, u, R, 1], ∇^ℓ_α[t, u, R, 1] ←_⊕ ν ⊗ β[s, u, R, 1] ⊗ α[t, u, R, 1]
      if s > 1 then
        ν ← ∇^ℓ_β[s, t, L, 0] ⊗ β[s, t, L, 0]
        for u = 1, ..., s do
          ∇^ℓ_β[u, t, L, 1], ∇^ℓ_α[u, s, L, 1] ←_⊕ ν ⊗ β[u, t, L, 1] ⊗ α[u, s, L, 1]
        ν ← ∇^ℓ_β[s, t, L, 1] ⊗ β[s, t, L, 1]
        for u = t, ..., n do
          ∇^ℓ_β[s, u, L, 1], ∇^ℓ_α[t, u, L, 0] ←_⊕ ν ⊗ β[s, u, L, 1] ⊗ α[t, u, L, 1]
        for u = 1, ..., s − 1 do
          γ ← β[u, t, R, 0] ⊗ α[u, s − 1, R, 1] ⊗ θ_ut
          ∇^ℓ_β[u, t, R, 0], ∇^ℓ_α[u, s − 1, R, 1], log ∇^ℓ_θ[u, t] ←_⊕ ν ⊗ γ
          γ ← β[u, t, L, 0] ⊗ α[u, s − 1, R, 1] ⊗ θ_tu
          ∇^ℓ_β[u, t, L, 0], ∇^ℓ_α[u, s − 1, R, 1], log ∇^ℓ_θ[t, u] ←_⊕ ν ⊗ γ
      ν ← ∇^ℓ_β[s, t, R, 1] ⊗ β[s, t, R, 1]
      for u = 1, ..., s do
        ∇^ℓ_β[u, t, R, 1], ∇^ℓ_α[u, s, R, 0] ←_⊕ ν ⊗ β[u, t, R, 1] ⊗ α[u, s, R, 0]
      for u = t + 1, ..., n do
        γ ← β[s, u, R, 0] ⊗ α[t + 1, u, L, 1] ⊗ θ_su
        ∇^ℓ_β[s, u, R, 0], ∇^ℓ_α[t + 1, u, L, 1], log ∇^ℓ_θ[s, u] ←_⊕ ν ⊗ γ
        γ ← β[s, u, L, 0] ⊗ α[t + 1, u, L, 1] ⊗ θ_us
        ∇^ℓ_β[s, u, L, 0], ∇^ℓ_α[t + 1, u, L, 1], log ∇^ℓ_θ[u, s] ←_⊕ ν ⊗ γ
  for k = n, ..., 1 do                               ▷ Backpropagate through inside step
    for s = 1, ..., n − k do
      t ← s + k
      ν ← ∇^ℓ_α[s, t, R, 1] ⊗ α[s, t, R, 1]
      for u = s + 1, ..., t do
        ∇^ℓ_α[u, t, R, 0], ∇^ℓ_α[u, t, R, 1] ←_⊕ ν ⊗ α[s, u, R, 0] ⊗ α[u, t, R, 1]
      if s > 1 then
        ν ← ∇^ℓ_α[s, t, L, 1] ⊗ α[s, t, L, 1]
        for u = s, ..., t − 1 do
          ∇^ℓ_α[s, u, L, 1], ∇^ℓ_α[u, t, L, 0] ←_⊕ ν ⊗ α[s, u, L, 1] ⊗ α[u, t, L, 0]
        ν ← ∇^ℓ_α[s, t, L, 0] ⊗ α[s, t, L, 0]
        for u = s, ..., t − 1 do
          γ ← α[s, u, R, 1] ⊗ α[u + 1, t, L, 1] ⊗ θ_ts
          ∇^ℓ_α[s, u, R, 1], ∇^ℓ_α[u + 1, t, L, 1], log ∇^ℓ_θ[t, s] ←_⊕ ν ⊗ γ
      ν ← ∇^ℓ_α[s, t, R, 0] ⊗ α[s, t, R, 0]
      for u = s, ..., t − 1 do
        γ ← α[s, u, R, 1] ⊗ α[u + 1, t, L, 1] ⊗ θ_st
        ∇^ℓ_α[s, u, R, 1], ∇^ℓ_α[u + 1, t, L, 1], log ∇^ℓ_θ[s, t] ←_⊕ ν ⊗ γ
  return signexp log ∇^ℓ_θ                           ▷ Exponentiate log gradient, multiply by sign, and return ∇^ℓ_θ
```

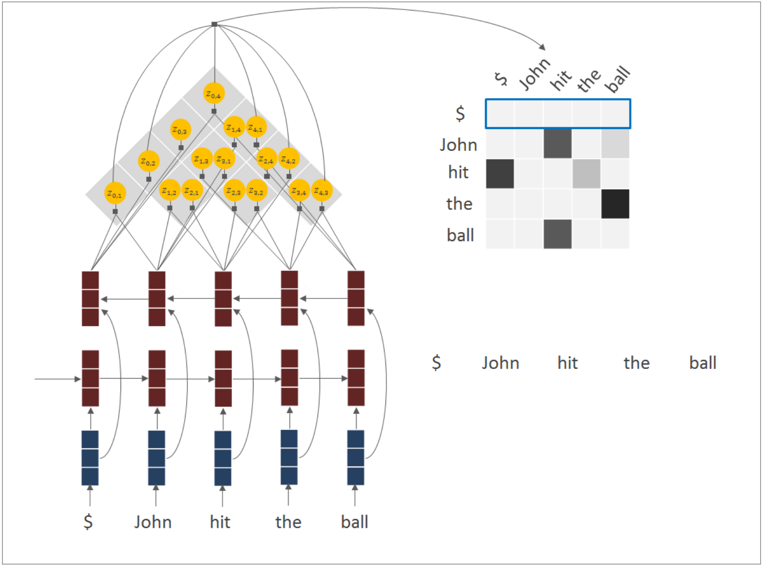
Syntactic Attention



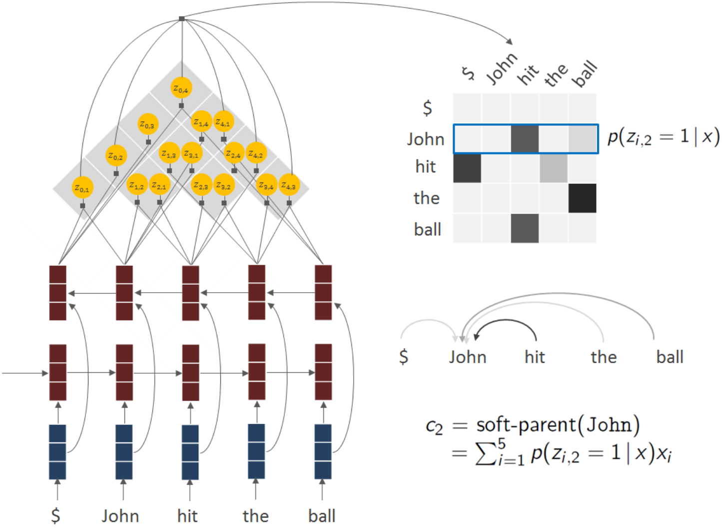
Syntactic Attention



Syntactic Attention

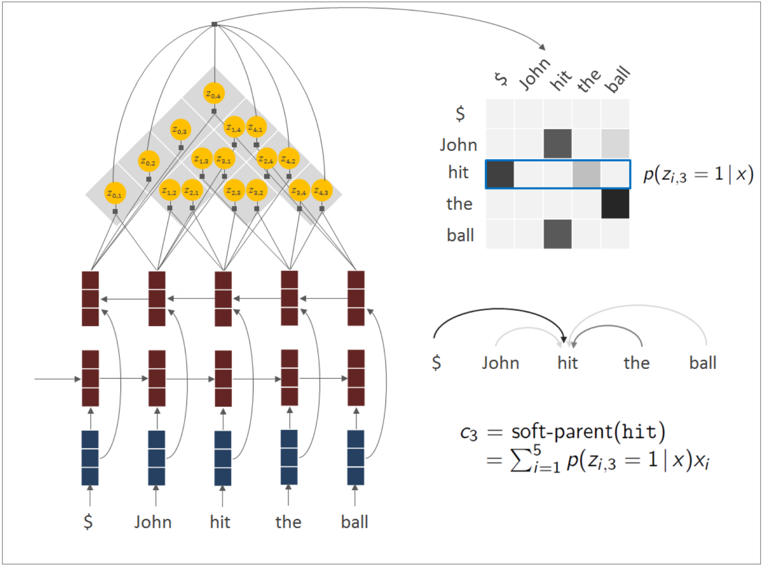


Syntactic Attention

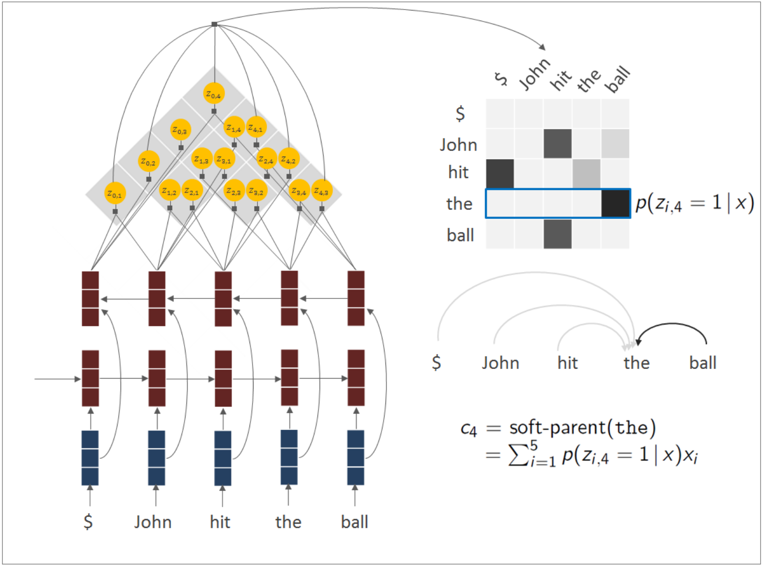


$$
\begin{aligned}
c_2 &= \text{soft-parent}(\text{John}) \\
&= \sum_{i=1}^{5} p(z_{i,2} = 1 \mid x) x_i
\end{aligned}
$$

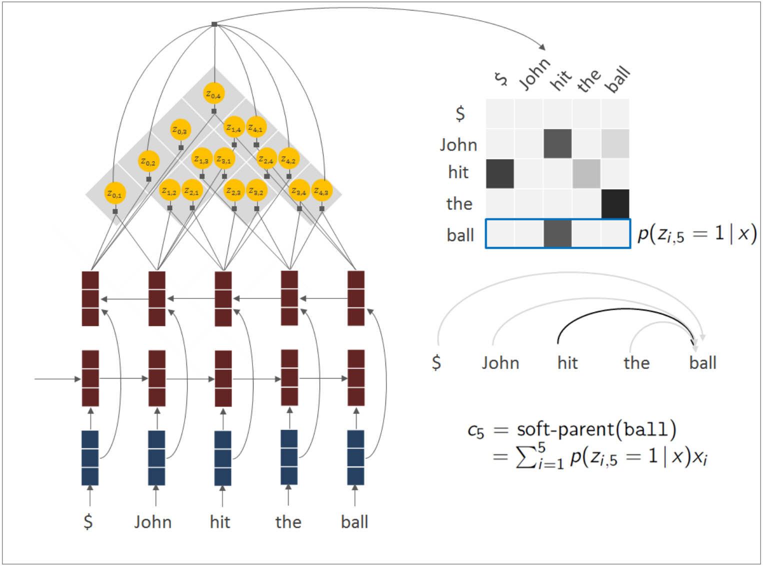
Syntactic Attention



Syntactic Attention



Syntactic Attention



Syntactic Attention for Natural Language Inference

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

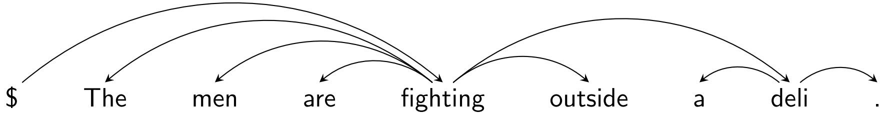
Model	Accuracy %
No Attention	85.8
Hard parent	86.1
Simple Attention	86.2
Structured Attention	86.8

- No attention: word embeddings only
- "Hard" parent from a pipelined dependency parser
- Simple attention (simple softmax instead of syntanctic 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:

$$\hat{z} = \arg\max_{z} p(z \mid x, q)$$



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 I

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Bowman, S. R., Manning, C. D., and Potts, C. (2015). Tree-Structured Composition in Neural Networks without Tree-Structured Architectures. In *Proceedings of the NIPS workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2015). Listen, Attend and Spell. *arXiv:1508.01211*.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-Based Models for Speech Recognition. In *Proceedings of NIPS*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

References II

Deng, Y., Kanervisto, A., and Rush, A. M. (2016). What You Get Is What You See: A Visual Markup Decompiler. *arXiv:1609.04938*.

Durrett, G. and Klein, D. (2015). Neural CRF Parsing. In *Proceedings of ACL*.

Eisner, J. M. (1996). Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of ACL*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. *arXiv:1410.5401*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature*.

References III

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In *Proceedings of NIPS*.

Kipperwasser, E. and Goldberg, Y. (2016). Simple and Accurate Dependency Parsing using Bidirectional LSTM Feature Representations. In *TACL*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based Learning Applied to Document Recognition. In *Proceedings of IEEE*.

Li, Z. and Eisner, J. (2009). First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of EMNLP 2009*.

References IV

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*.

Parikh, A. P., Tackstrom, O., Das, D., and Uszkoreit, J. (2016). A Decomposable Attention Model for Natural Language Inference. In *Proceedings of EMNLP*.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kocisky, T., and Blunsom, P. (2016). Reasoning about Entailment with Neural Attention. In *Proceedings of ICLR*.

Rush, A. M., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-To-End Memory Networks. In *Proceedings of NIPS*.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*.

References V

Vinyals, O., Fortunato, M., and Jaitly, N. (2015a). Pointer Networks. In *Proceedings of NIPS*.

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015b). Grammar as a Foreign Language. In *Proceedings of NIPS*.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards Ai-complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv preprint arXiv:1502.05698*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICML*.