CSCI 544: Applied Natural Language Processing

Seq2seq Generation & Neural Machine Translation

Xuezhe Ma (Max)

USC University of Southern California

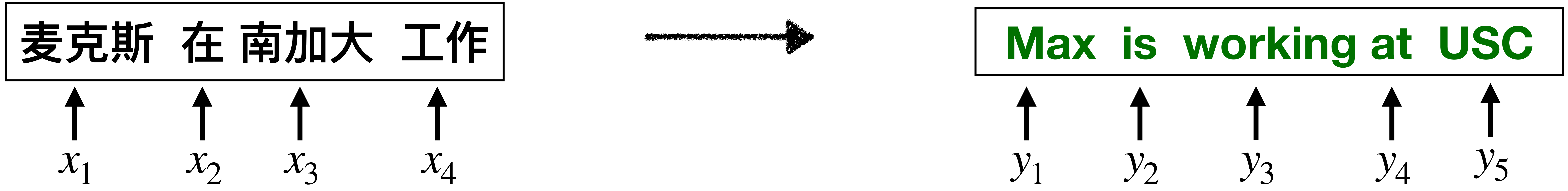
Recap: Statistical Machine Translation

- **IBM Translation Models**
  - Word-level alignment model
  - EM algorithm
- **Phrase-based Translation Models**
  - Phrase-based alignment model
- **Heavy Engineering**
  - Moses system
  - 360 pages manual

Neural Machine Translation

Seq2seq Generation

- **Sequence-to-Sequence (Seq2seq) Generation**
  - Input: $X = \{x_1, x_2, \ldots, x_L\}, x_i \in \mathcal{X}$
  - Output: $Y = \{y_1, y_2, \ldots, y_T\}, y_i \in \mathcal{Y}$
  - Model: $p_\theta(Y|X)$



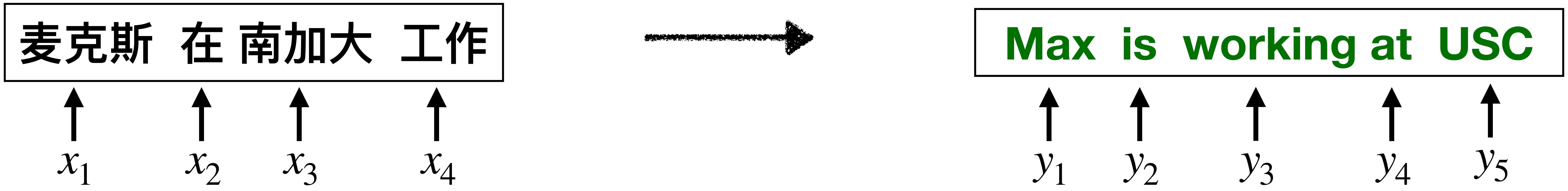
Seq2seq Generation

- **Sequence-to-Sequence (Seq2seq) Generation**
  - Input: $X = \{x_1, x_2, \ldots, x_L\}, x_i \in \mathcal{X}$
  - Output: $Y = \{y_1, y_2, \ldots, y_T\}, y_i \in \mathcal{Y}$
  - Model: $p_\theta(Y|X)$

Input X	Output Y (**Text**)	Task
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

Seq2seq Generation

- **Sequence-to-Sequence (Seq2seq) Generation**
  - Input: $X = \{x_1, x_2, \ldots, x_L\}, x_i \in \mathcal{X}$
  - Output: $Y = \{y_1, y_2, \ldots, y_T\}, y_i \in \mathcal{Y}$
  - Model: $p_\theta(Y|X)$ **How?**
- **Difference from Sequence Labeling**
  - The length of Y can be different from the length of X
  - The size of $\mathcal{Y}$ is often much larger



Autoregressive Seq2seq Generation

- **Autoregressive Factorization:**

$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(\underbrace{y_t}_{\text{Next Token}} | \underbrace{y_{\boxed{<t}}}_{\text{history}}, X)$$

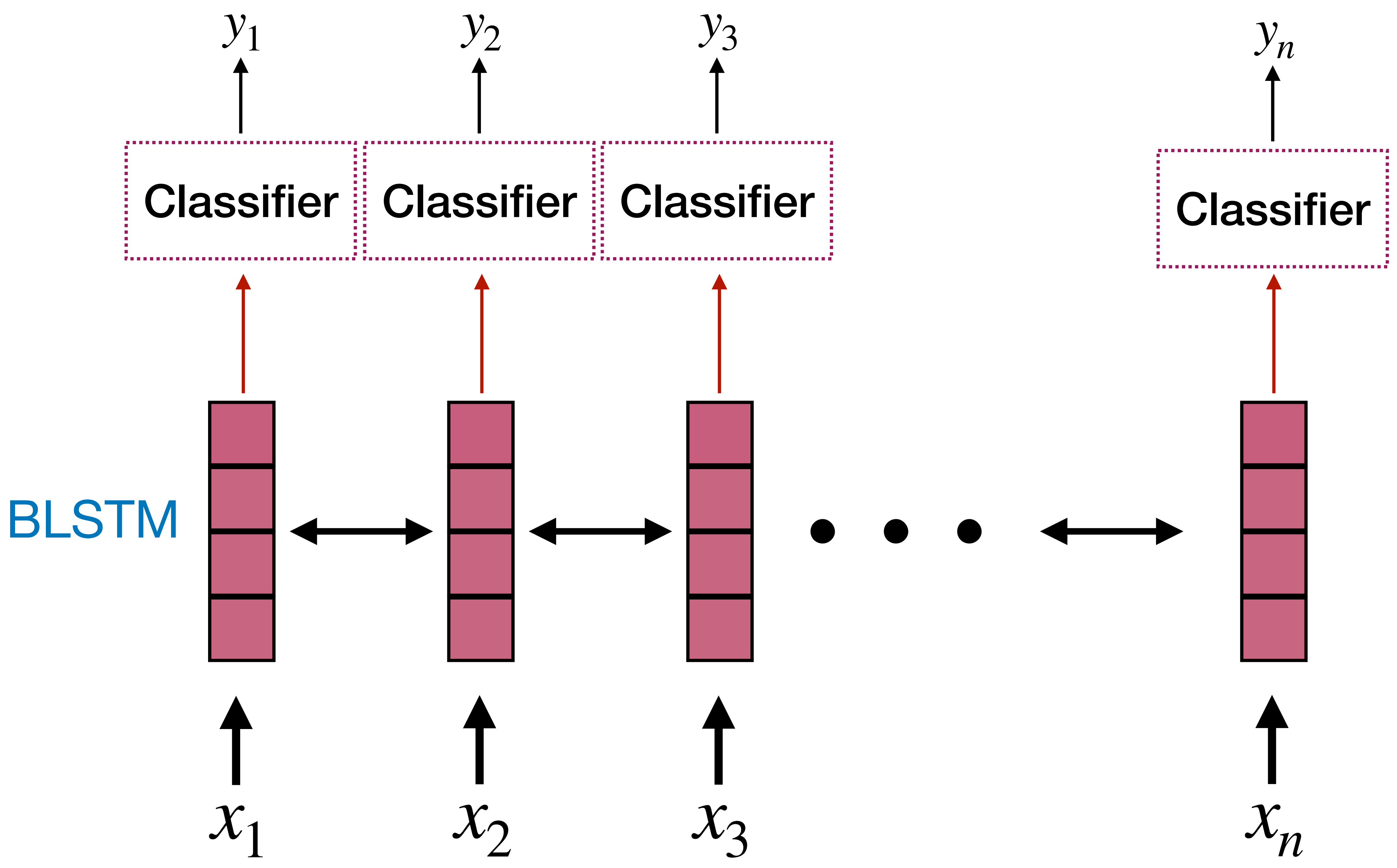
- Autoregressive factorization is just chain-rule (HMMs, MEMMs)
- Autoregressive factorization does NOT assume any independence
- With autoregressive factorization, we need to model each $p_\theta(y_t | y_{<t}, X)$

Autoregressive Seq2seq Generation

- **Sequence labeling vs. Seq2seq Generation**

Sequence labeling

$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t|X)$$



Why not for seq2seq generation?

Autoregressive Seq2seq Generation

$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t|X)$$

Not a good choice!

我 不 知道

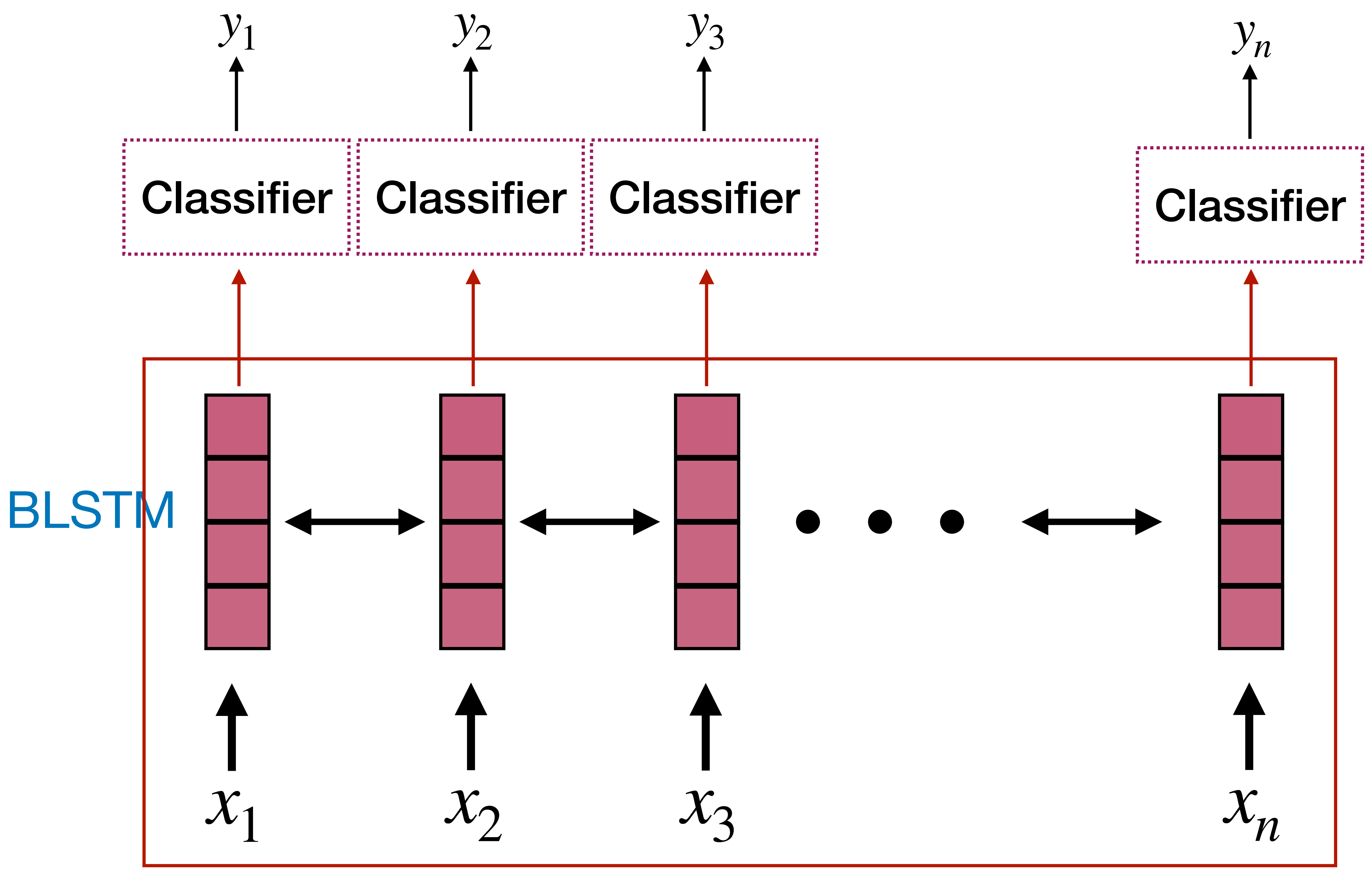
I	don't	know	
I	do	not	know
I	have	no	idea

Encoder-Decoder Architecture

- **Sequence labeling vs. Seq2seq Generation**

Sequence labeling

$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t|\boxed{X})$$



Seq2seq Generation

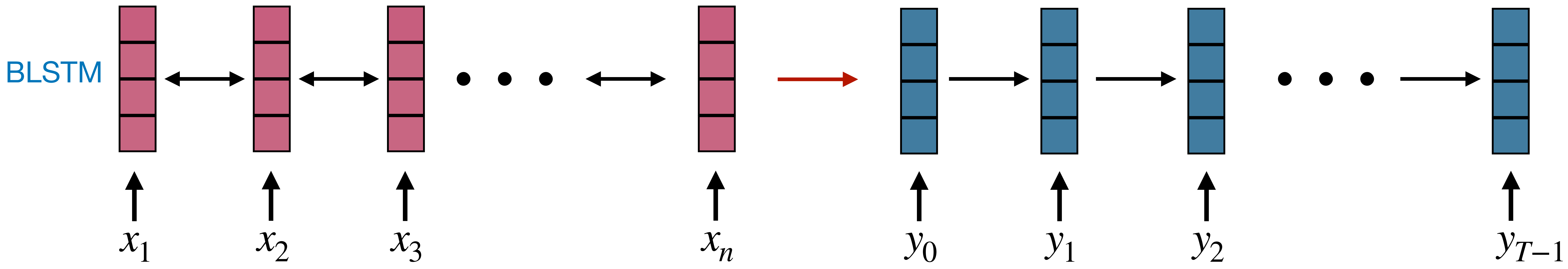
$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(\boxed{y_t|y_{<t}}, \boxed{X})$$

Encoder: encode a sentence into a sequence of vectors

Decoder: use another LSTM?

Encoder-Decoder Architecture

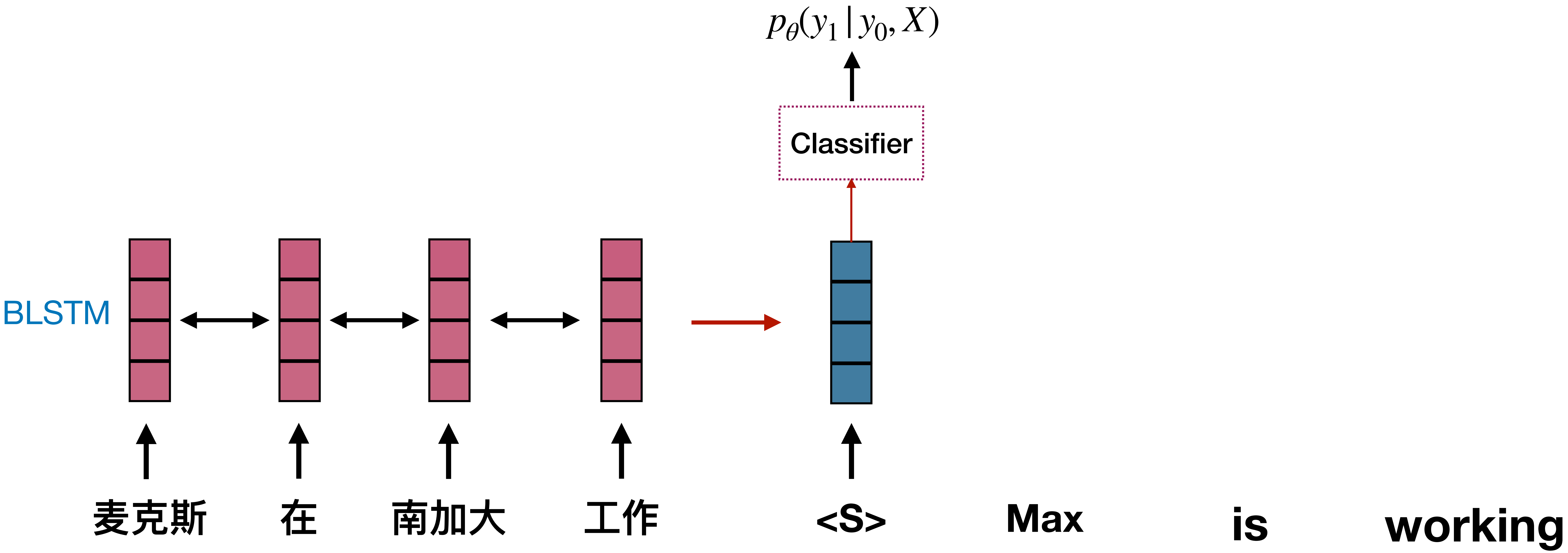
- **Two Components:**
  - Encoder: Convert input sequence into a sequence of vectors
  - Decoder: Convert encoding into a sequence in the output space



Seq2seq Training

- **Model Training:**

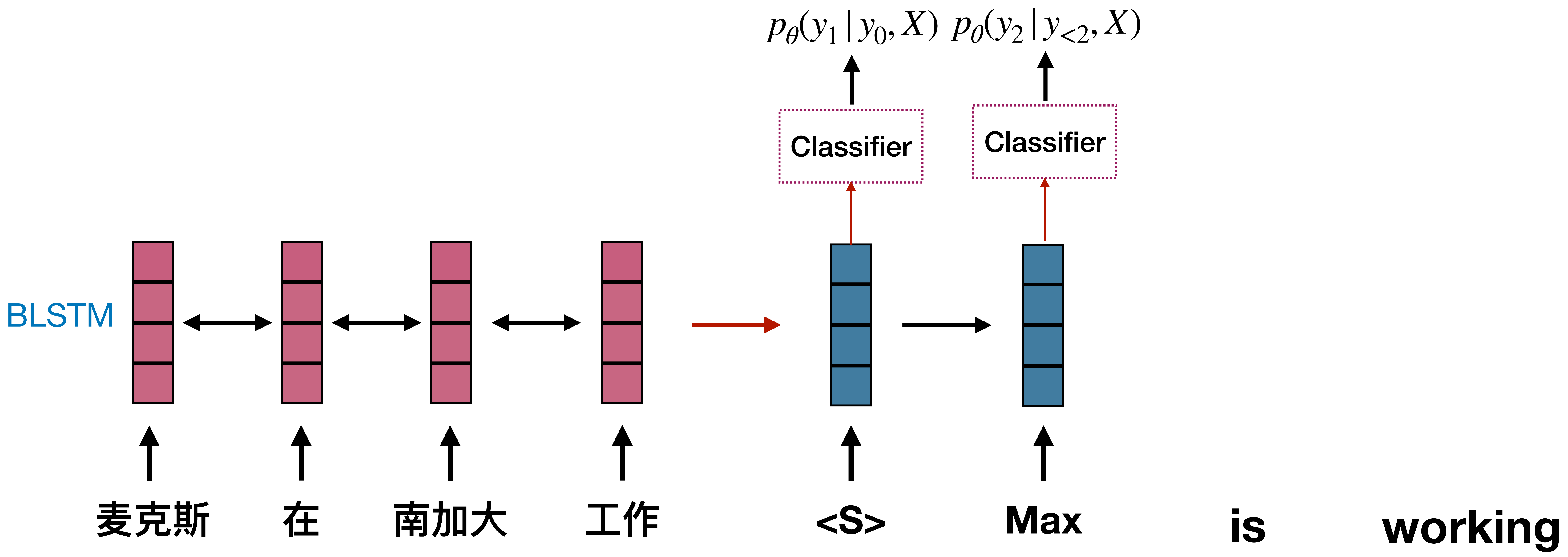
$$p_\theta(Y|X) = \prod_{t=1}^{T} \boxed{p_\theta(y_t|y_{<t}, X) \quad t=1}$$



Seq2seq Training

- **Model Training:**

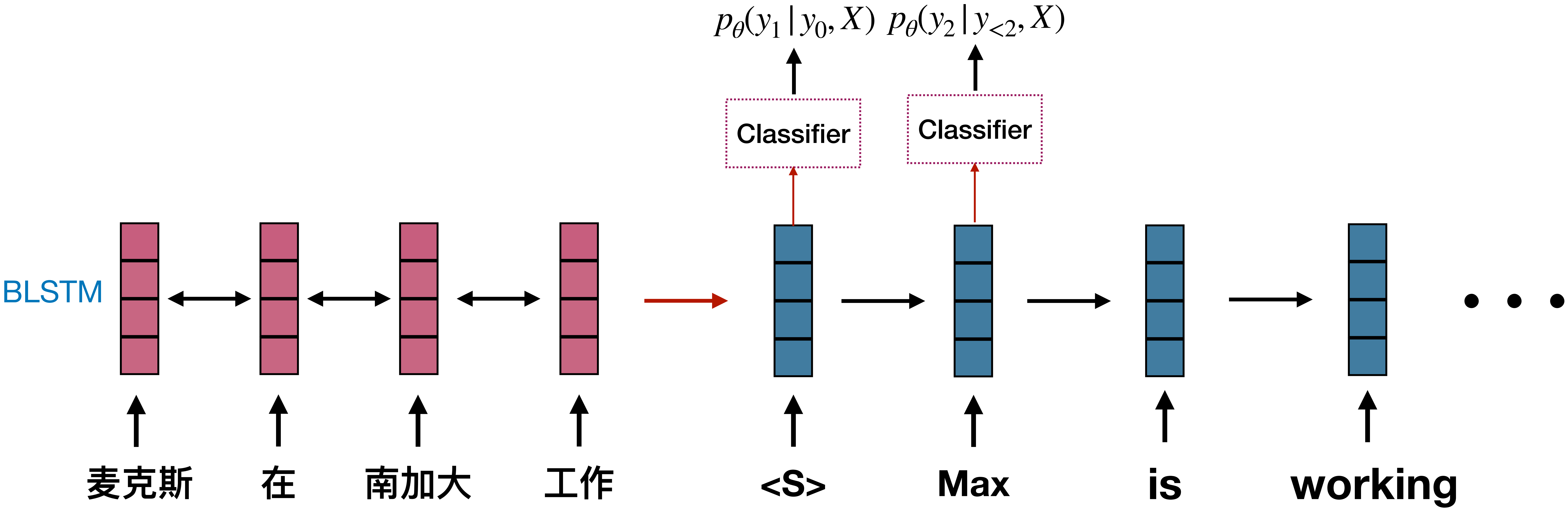
$$p_\theta(Y|X) = \prod_{t=1}^{T} \boxed{p_\theta(y_t|y_{<t}, X) \quad t = 2}$$



Seq2seq Training

- **Model Training:**

$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, X)$$



Seq2seq Training

- **Maximum Likelihood Estimation**

$$\max_{\theta} p_{\theta}(Y|X) = \prod_{t=1}^{T} p_{\theta}(y_t|y_{<t}, X)$$

- **Back-propagate gradients through both decoder & encoder**
- **Need a really big training corpus**
  - WMT Russian-English

Seq2seq Decoding

- **Exhaustive Search**
  - Requires computing all possible sequences

$$\arg\max_{Y} p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, X)$$

What is the complexity of doing this search, if $|\mathcal{Y}| = V$ and sequence length T?

(a) $O(VT)$

(b) $\boxed{O(V^T)}$

(c) $O(T^V)$

Seq2seq Decoding

- **Greedy Search**
  - Selects the best current word y_t

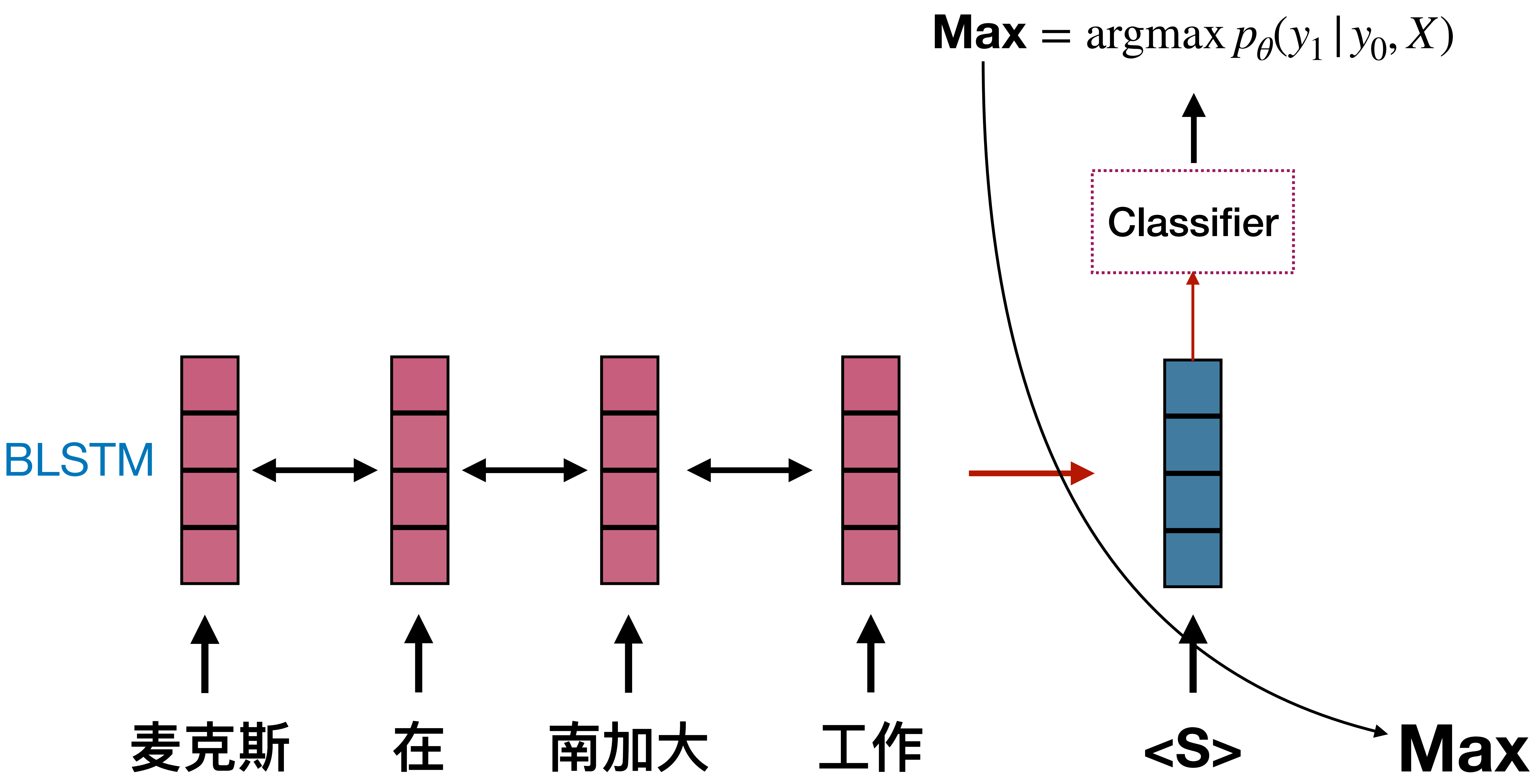
$$\arg\max_{Y} p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, X)$$

$$\approx \mathbf{arg}\max_{y_t} p_\theta(y_t | y_{<t}, X), \forall t$$

Seq2seq Decoding

- **Greedy decoding:**

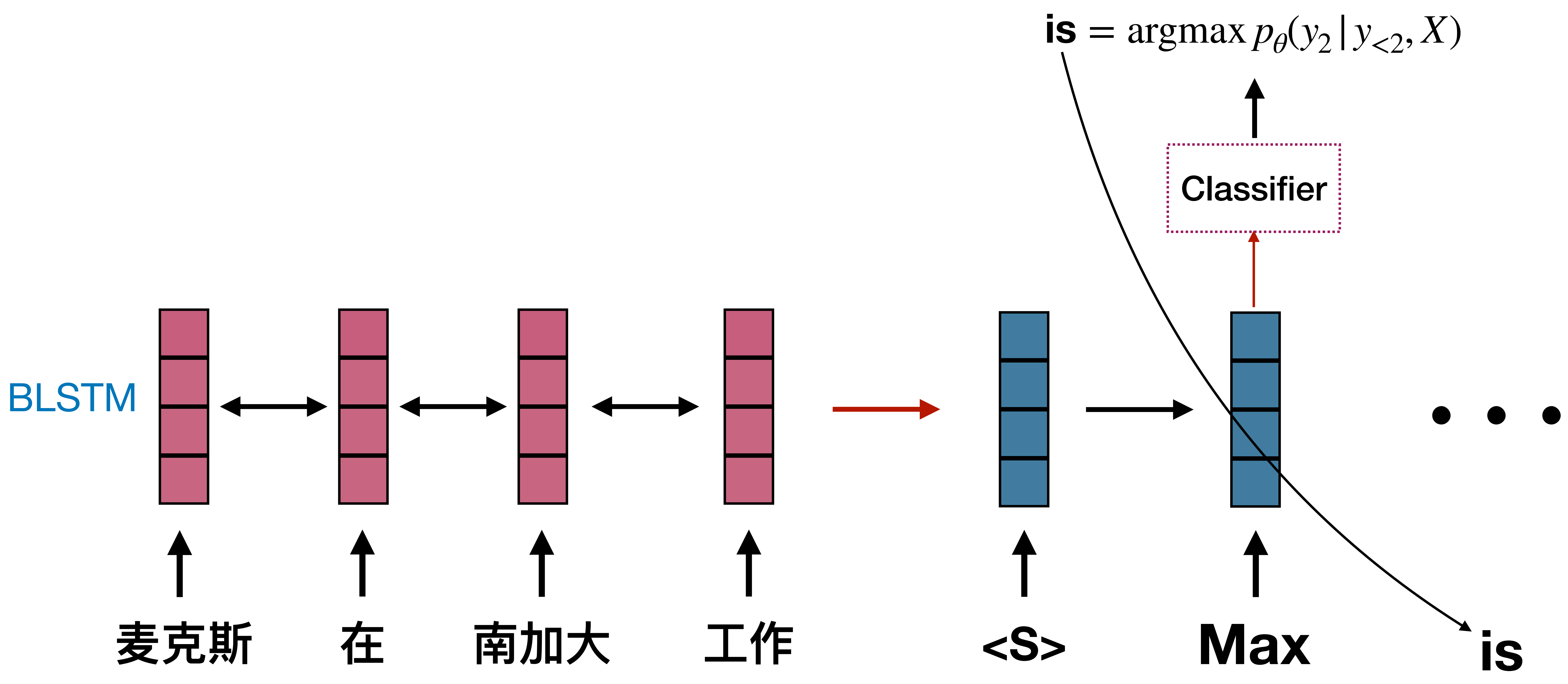
$$y_t{}^* = \arg\max_{y_t} p_\theta(y_t \mid y_{<t}, X), \forall t$$



Seq2seq Decoding

- **Greedy decoding:**

$$y_t{}^* = \arg\max_{y_t} p_\theta(y_t \mid y_{<t}\ , X), \forall t$$

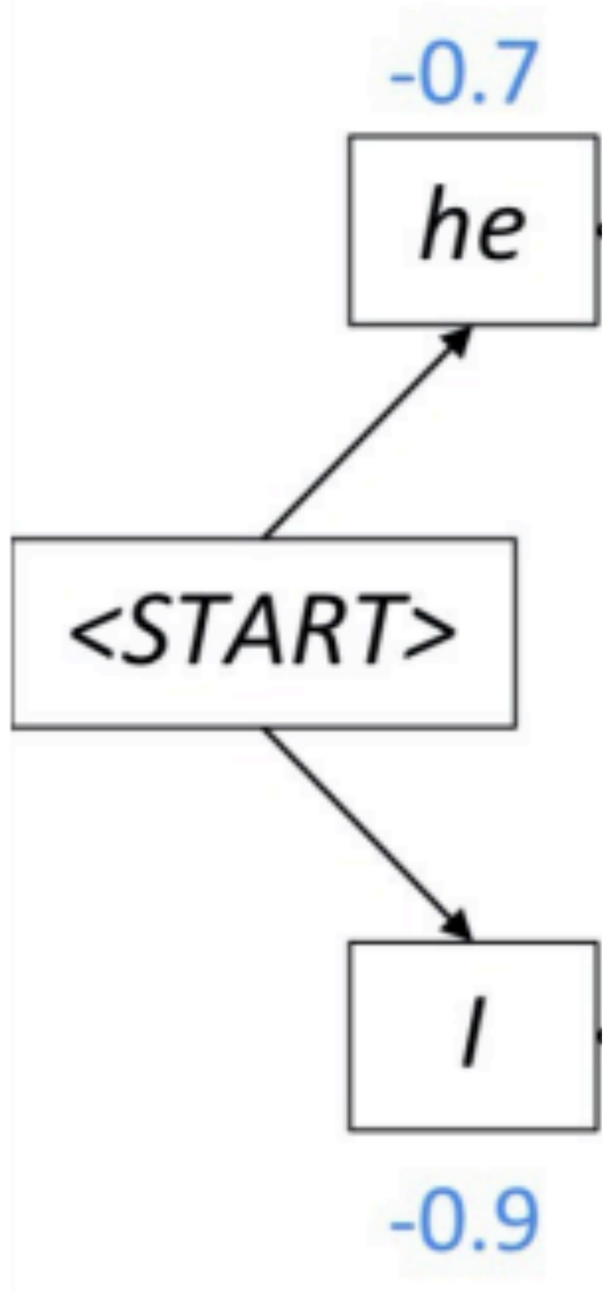


A Middle Ground: Beam Search

- **Key idea:** at every step, keep track of the k most probable partial translations (hypotheses)
- Score of each hypothesis = log probability of sequence so far
- Not guaranteed to be optimal
- More efficient than exhaustive search

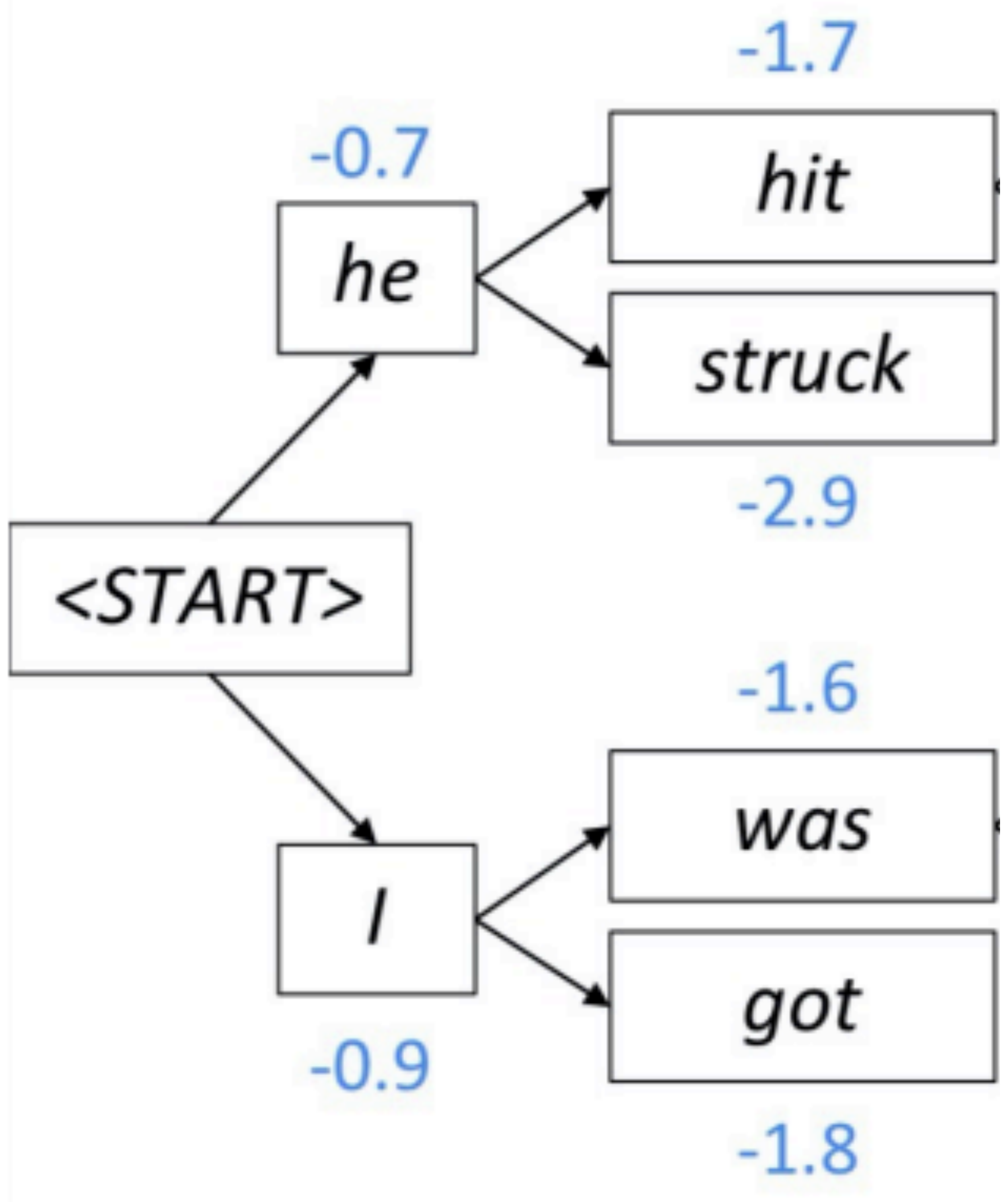
Beam Search Decoding

Beam size $K = 2$



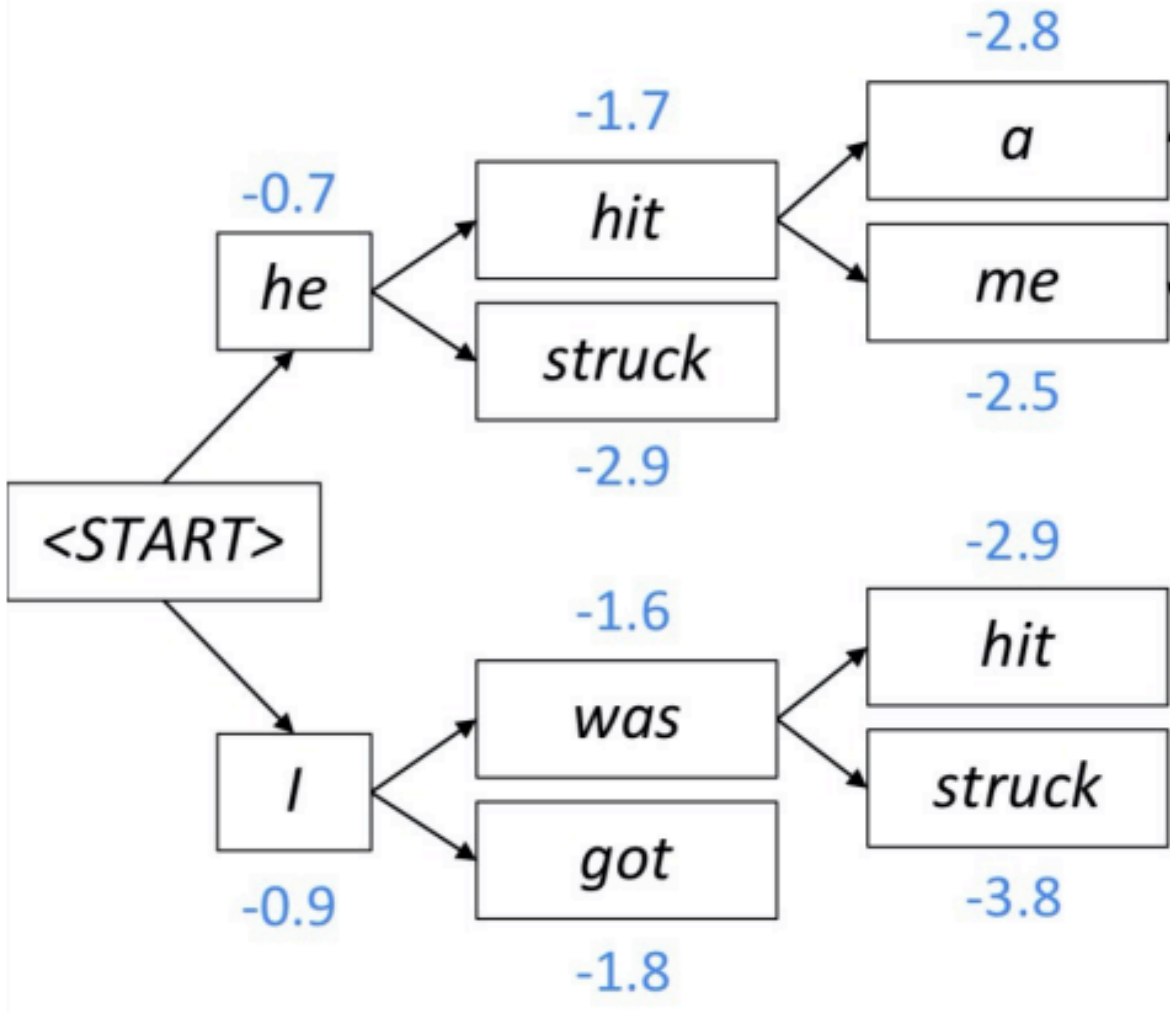
Beam Search Decoding

Beam size $K = 2$



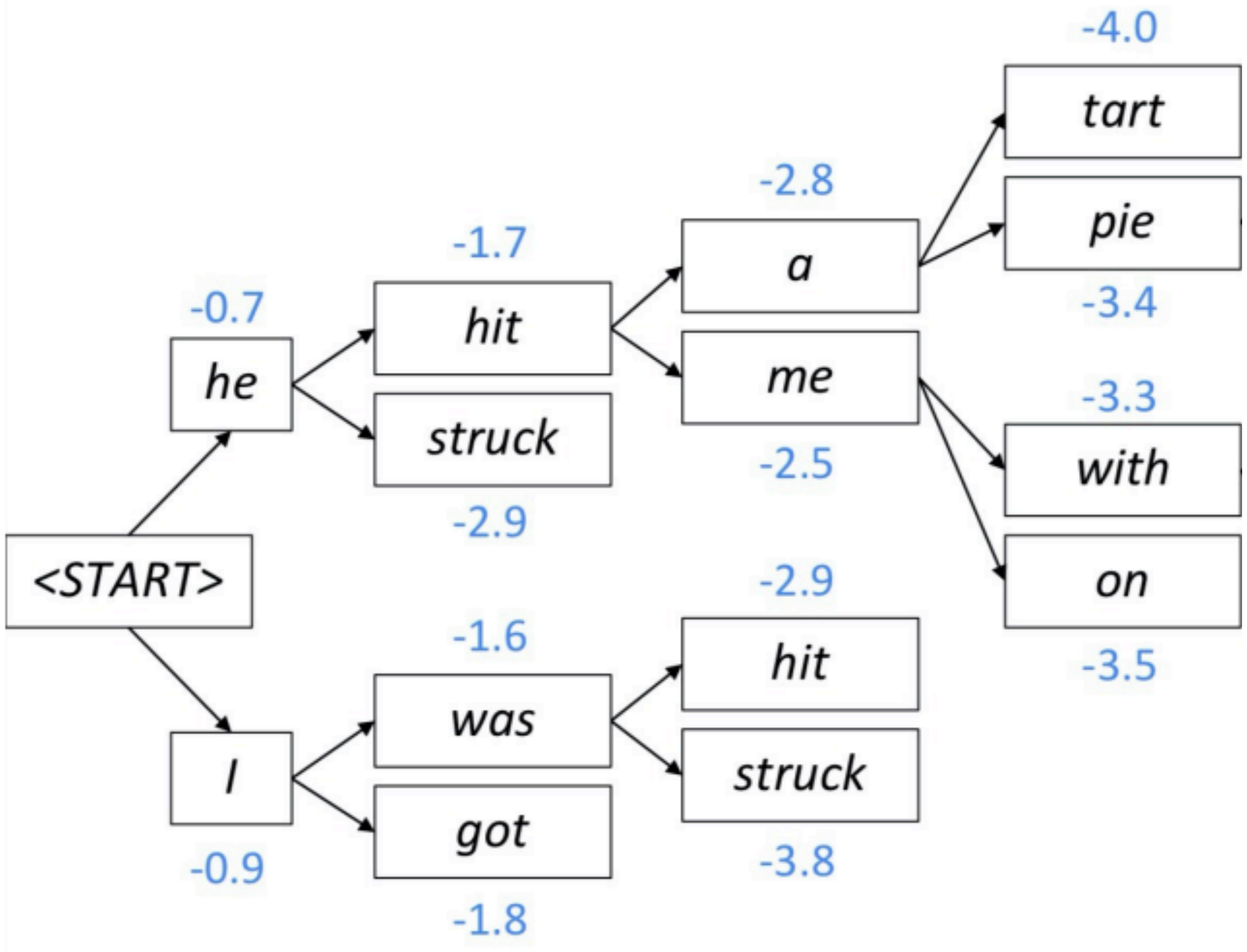
Beam Search Decoding

Beam size $K = 2$



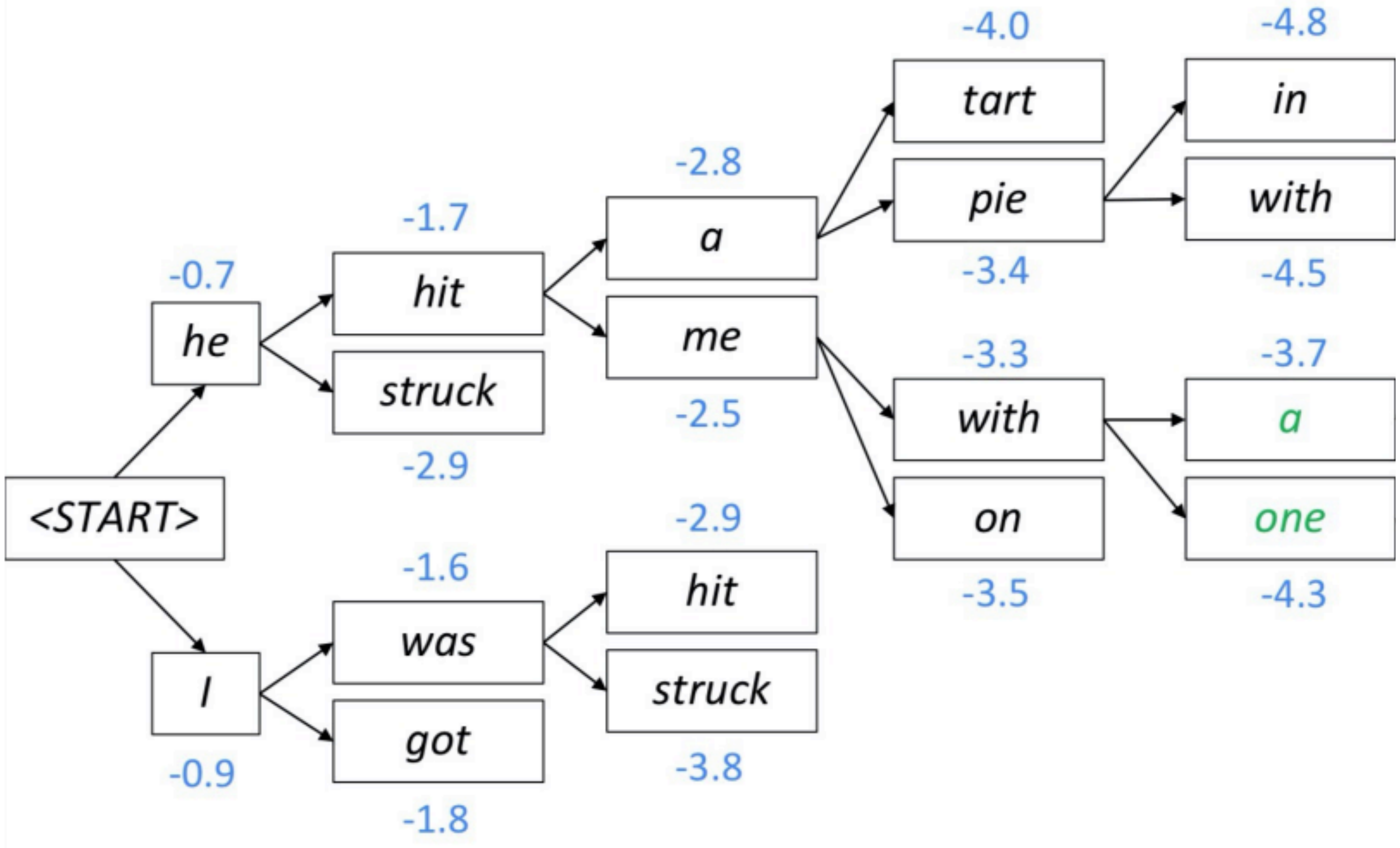
Beam Search Decoding

Beam size $K = 2$



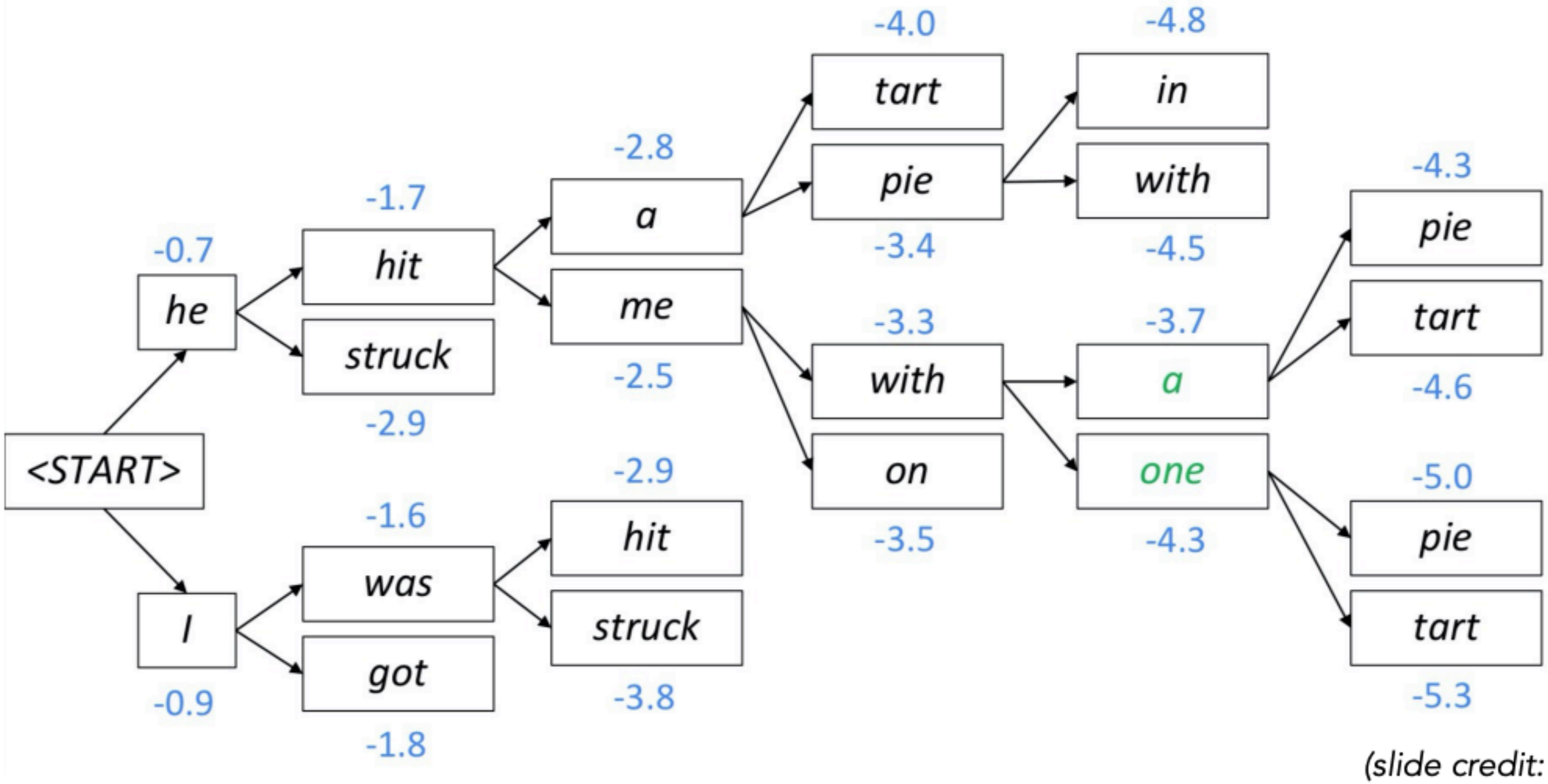
Beam Search Decoding

Beam size $K = 2$



Beam Search Decoding

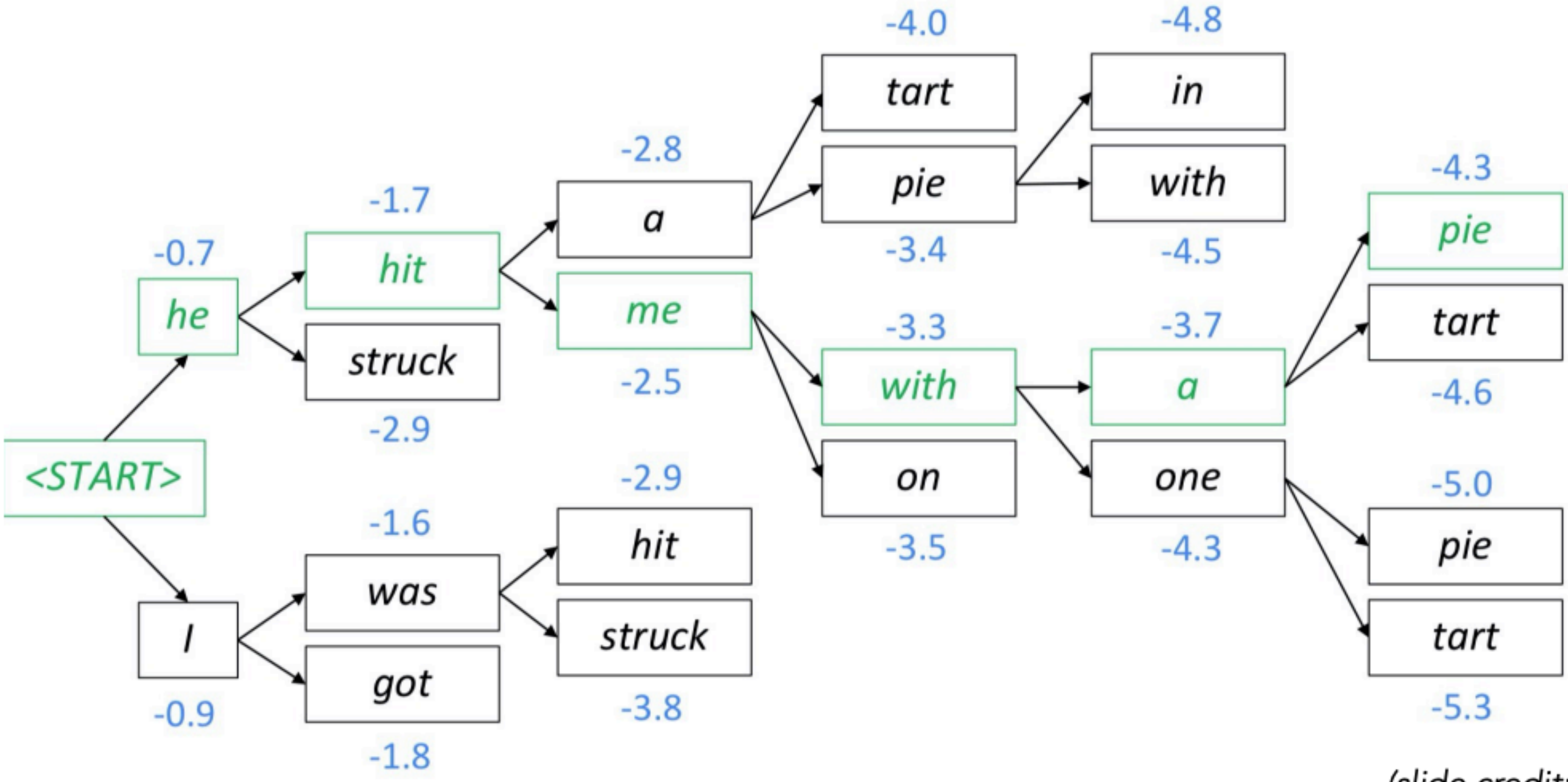
Beam size $K = 2$



(slide credit: Abigail See)

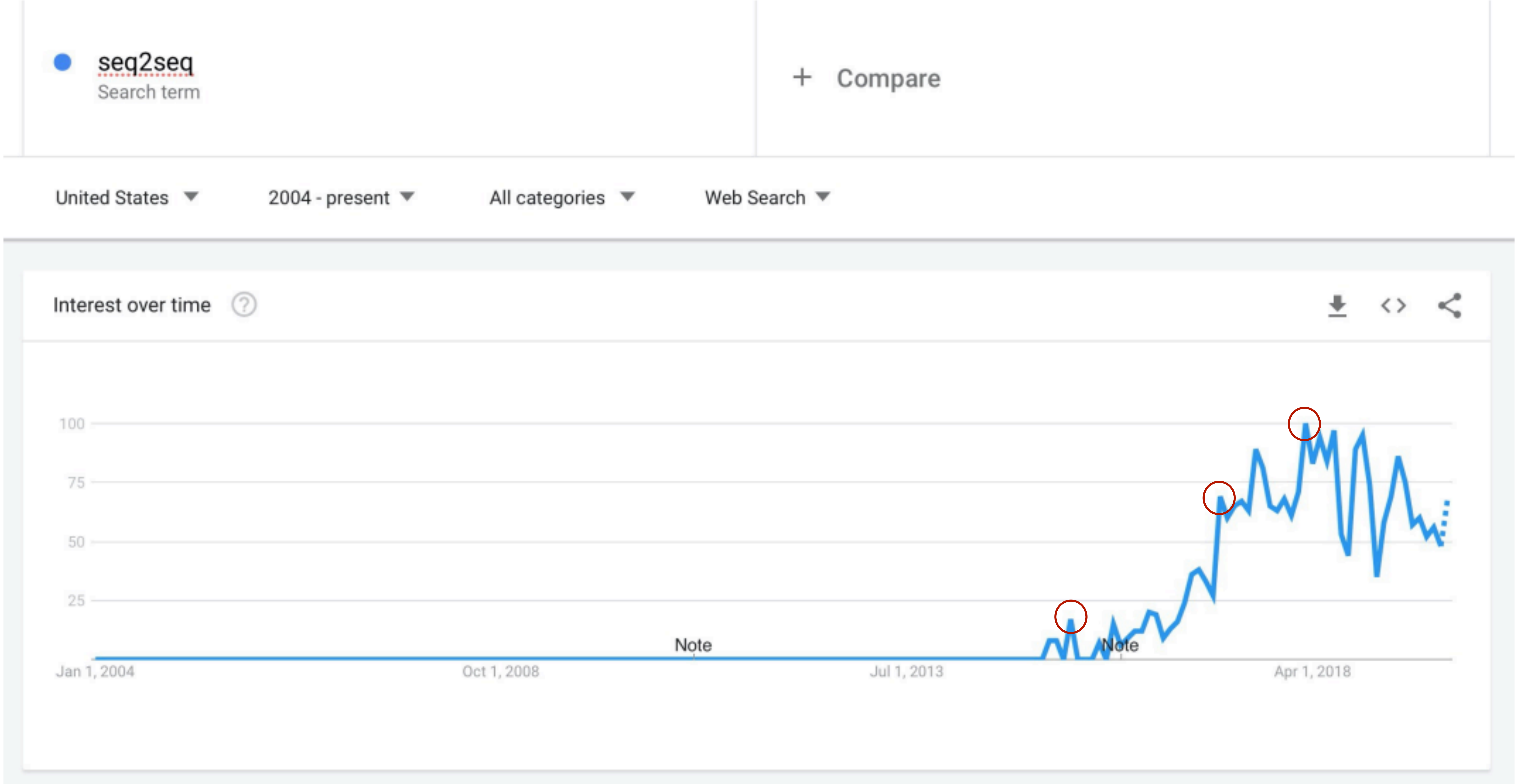
Backtrack

Beam size $K = 2$

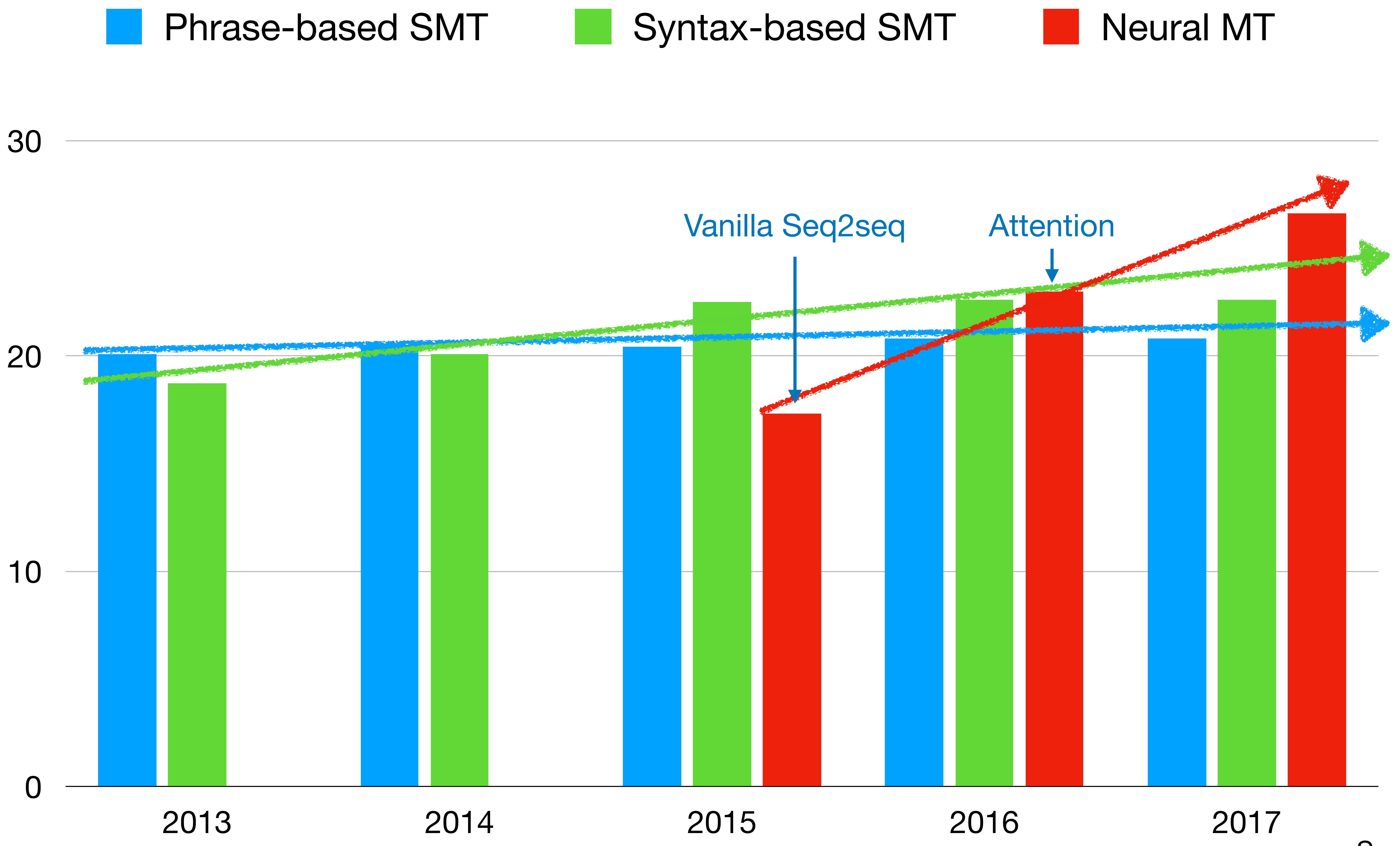


(slide credit: Abigail See)

How Seq2seq changed the MT Landscape



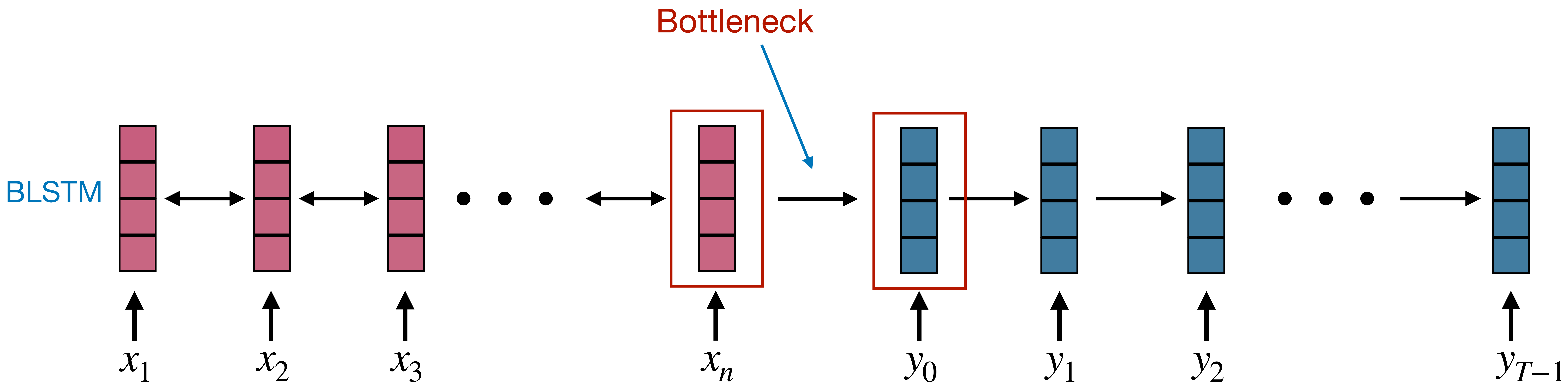
MT Progress



Source: Rico Sennrich

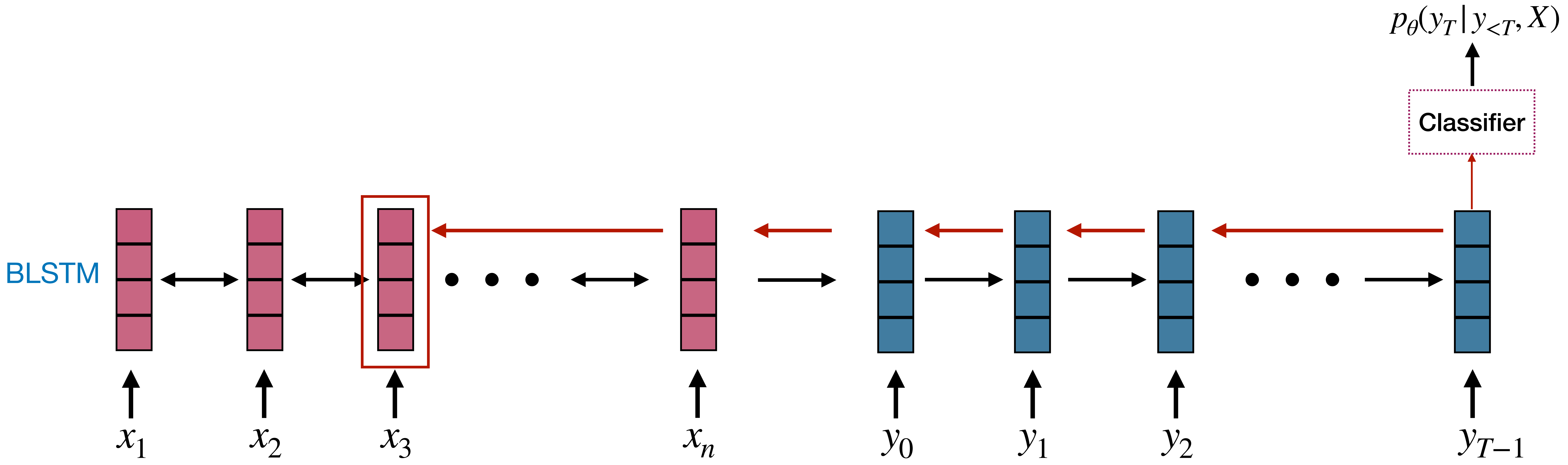
Issues with Vanilla Encoder-Decoder Architecture

- A single encoding vector needs to capture all the information about source sentence



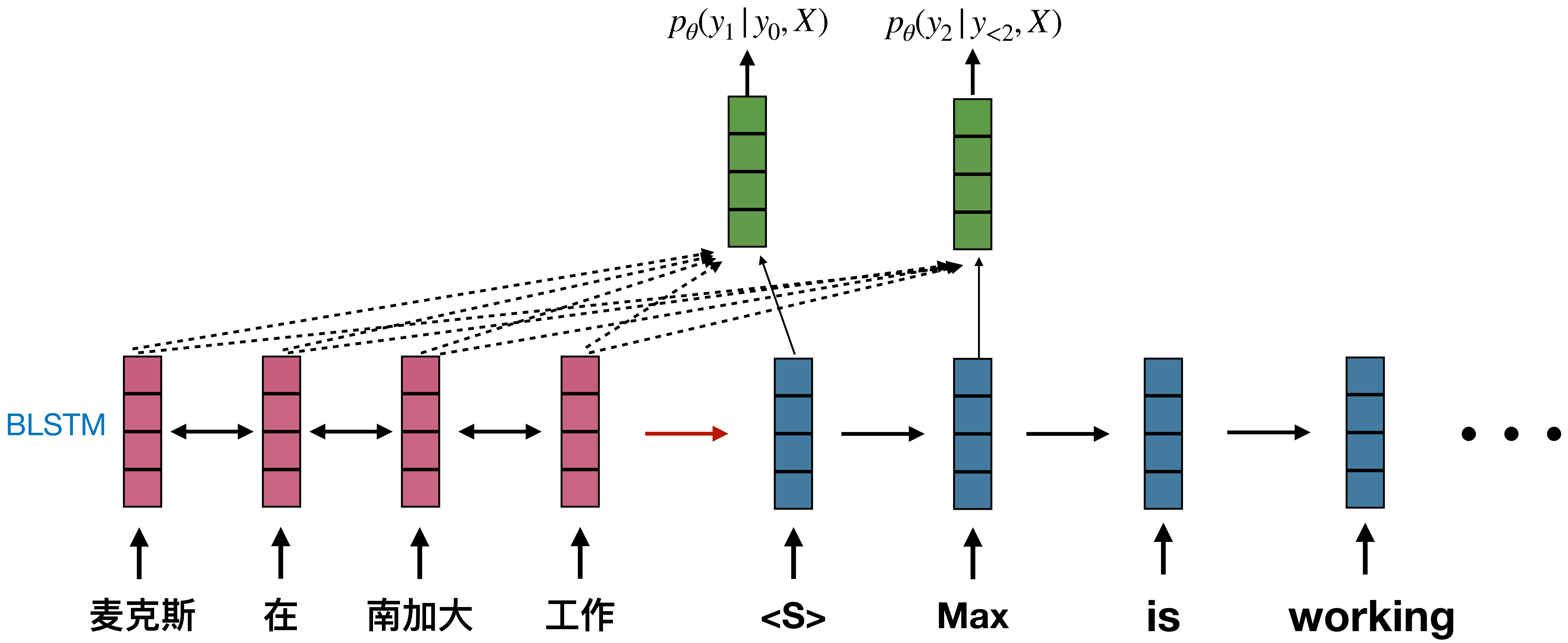
Issues with Vanilla Encoder-Decoder Architecture

- A single encoding vector needs to capture **all the information** about source sentence
- Longer sequences can lead to **vanishing gradients**

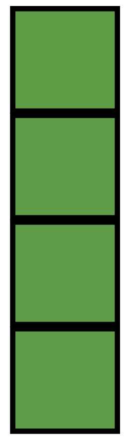


Attention Mechanism

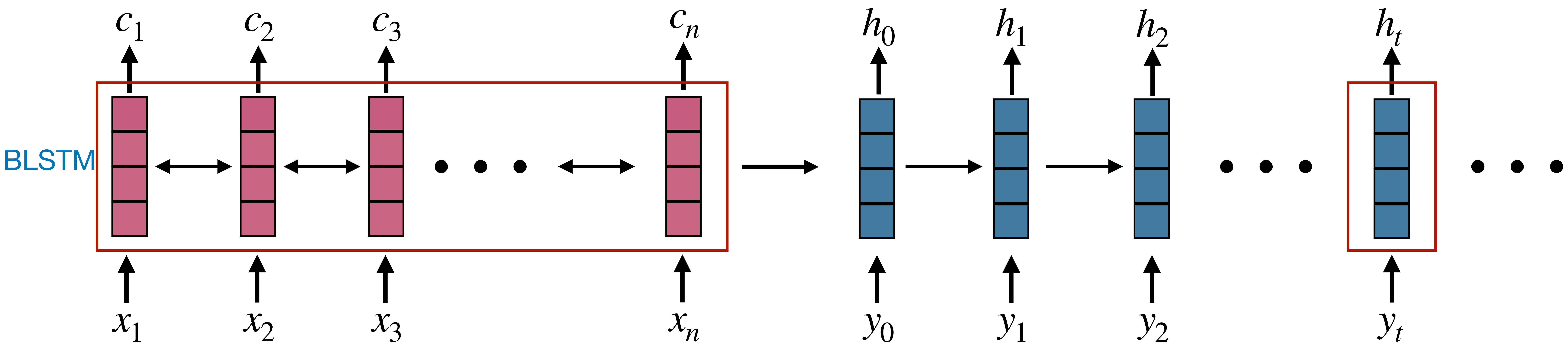
- **Key idea:** At each time step, use all parts of source sentence



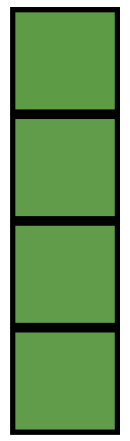
Attention Mechanism



$= \text{attn}([c_1, c_2, \ldots, c_n], h_t)$



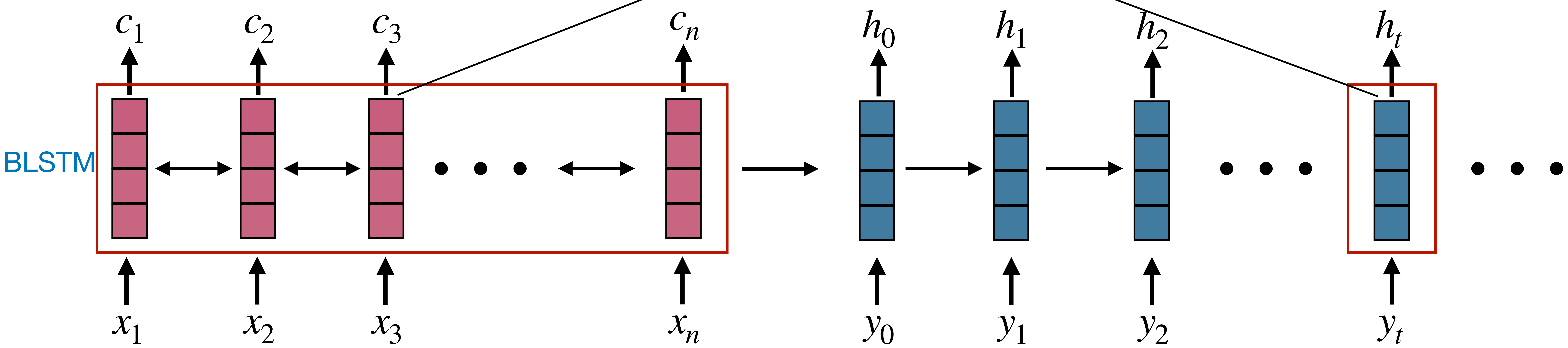
Attention Mechanism



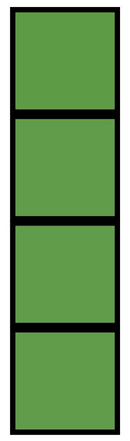
$$= \text{attn}([c_1, c_2, \ldots, c_n], h_t)$$

$$e_j^t = \text{sim}(c_j, h_t),\ \forall j \in \{1,\ldots,n\}$$

Attention scores



Attention Mechanism



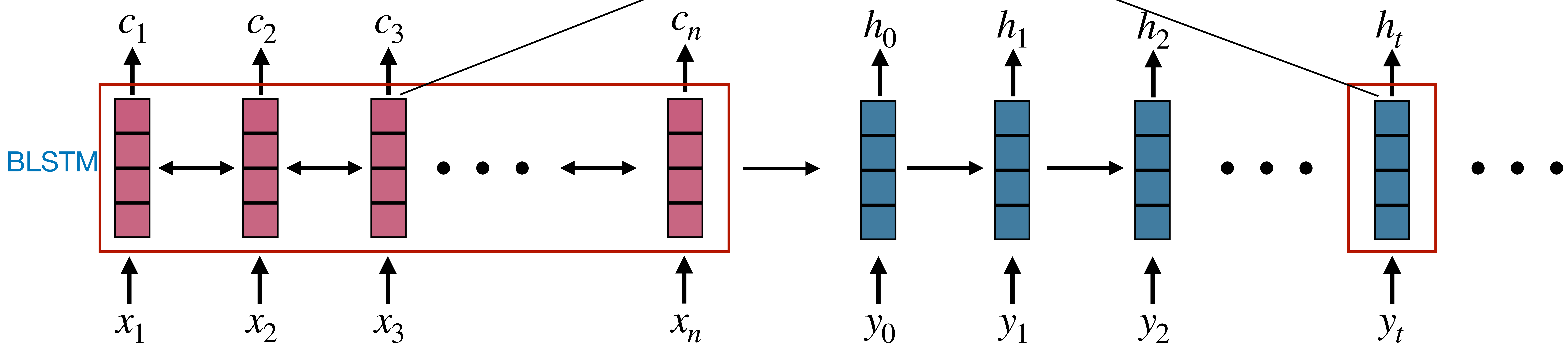
$$= \text{attn}([c_1, c_2, \ldots, c_n], h_t)$$

$$a^t = \text{softmax}(e^t) \in \mathbb{R}^n$$

Attention distribution

$$e^t_j = \text{sim}(c_j, h_t),\ \forall j \in \{1, \ldots, n\}$$

Attention scores



Attention Mechanism

$$= \mathrm{attn}([c_1, c_2, \ldots, c_n], h_t) = \sum_{j=1}^{n} a_j^t c_j \in \mathbb{R}^d$$

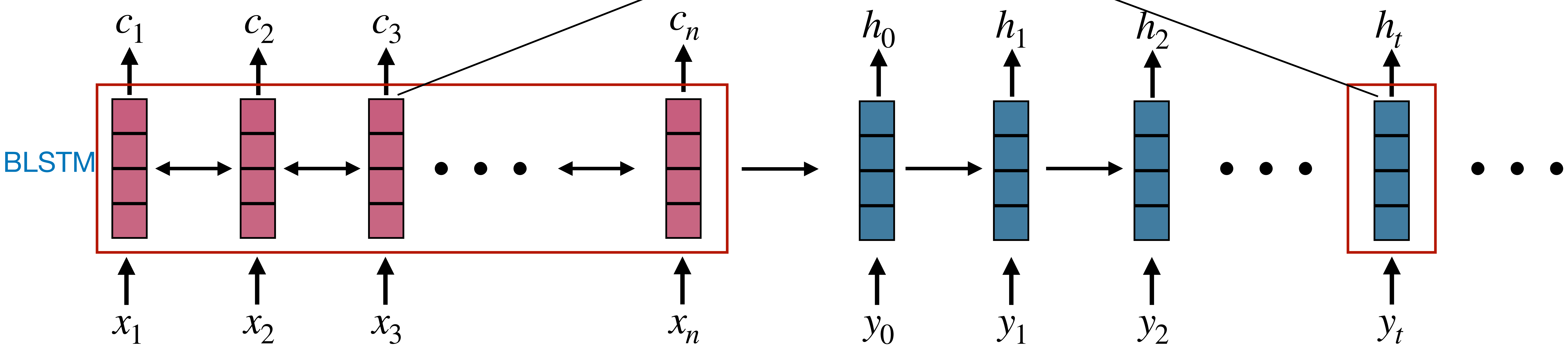
Attention output

$$a^t = \mathrm{softmax}(e^t) \in \mathbb{R}^n$$

Attention distribution

$$e_j^t = \mathrm{sim}(c_j, h_t), \ \forall j \in \{1, \ldots, n\}$$

Attention scores



Types of Attention

- **Dot-product attention** (assumes equal dimensions for c and h)

$$\mathrm{sim}(c_j, h_t) = c_j^T h_t$$

- **Multiplicative attention**

$\mathrm{sim}(c_j, h_t) = c_j^T W h_t$, where W is learnable weight matrix

- **Additive attention**

$$\mathrm{sim}(c_j, h_t) = v^T \tanh(W_c c_j + W_h h_t)$$

where W_c and W_h are learnable weight matrices and v is a learnable weight vector

Attention Improves Translation Performance

System	Ppl	BLEU
Winning WMT'14 system – *phrase-based* + *large LM* (Buck et al., 2014)		20.7
Existing NMT systems		
RNNsearch (Jean et al., 2015)		16.5
RNNsearch + unk replace (Jean et al., 2015)		19.0
RNNsearch + unk replace + large vocab + *ensemble* 8 models (Jean et al., 2015)		**21.6**
Our NMT systems		
Base	10.6	11.3
Base + reverse	9.9	12.6 (+*1.3*)
Base + reverse + dropout	8.1	14.0 (+*1.4*)
Base + reverse + dropout + global attention (*location*)	7.3	16.8 (+*2.8*)
Base + reverse + dropout + global attention (*location*) + feed input	6.4	18.1 (+*1.3*)
Base + reverse + dropout + local-p attention (*general*) + feed input	5.9	19.0 (+*0.9*)
Base + reverse + dropout + local-p attention (*general*) + feed input + unk replace		20.9 (+*1.9*)
Ensemble 8 models + unk replace		**23.0** (+*2.1*)

(Luong et al., 2015)

RESEARCH › PUBLICATIONS ›

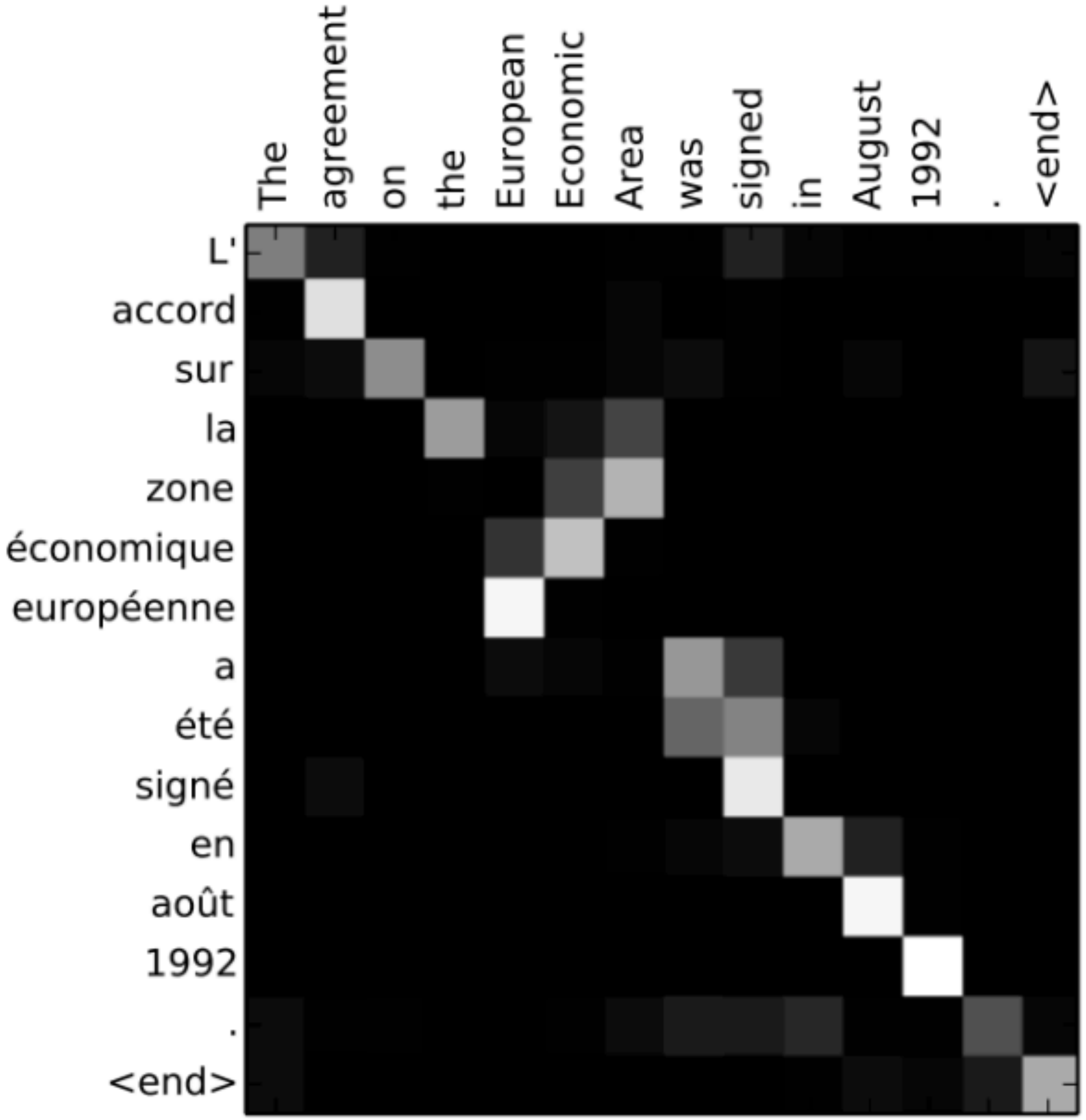
Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English $\rightarrow$ Spanish	4.885	5.428	5.504	87%
English $\rightarrow$ French	4.932	5.295	5.496	64%
English $\rightarrow$ Chinese	4.035	4.594	4.987	58%
Spanish $\rightarrow$ English	4.872	5.187	5.372	63%
French $\rightarrow$ English	5.046	5.343	5.404	83%
Chinese $\rightarrow$ English	3.694	4.263	4.636	60%

(Wu et al., 2016)

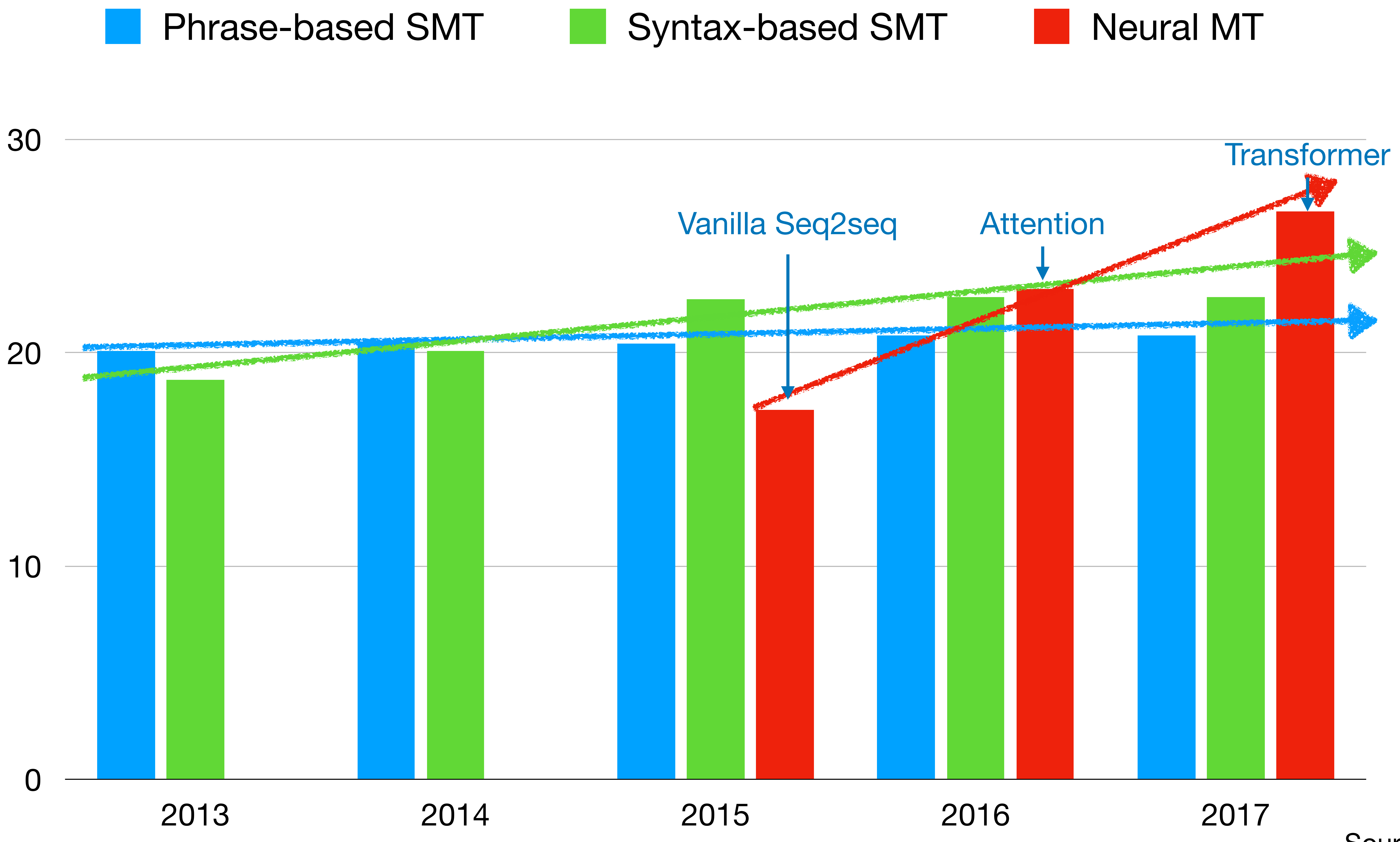
Visualizing Attention



Highly correlated with alignment

credits: Jay Alammar

MT Progress



Source: Rico Sennrich

Reading Materials

- **Reading Materials**
  - Sequence to Sequence Learning with Neural Networks
  - Neural Machine Translation by Jointly Learning to Align and Translate