CSCI 544: Applied Natural Language Processing

Seq2seq Generation
& Neural Machine Translation

Xuezhe Ma (Max)

USC University of Southern California

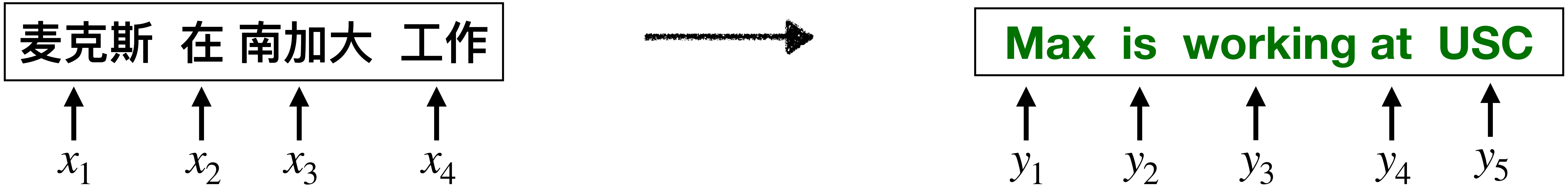
Course Organization

NLP Tasks

Models	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

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: $p_\theta(Y|X)$



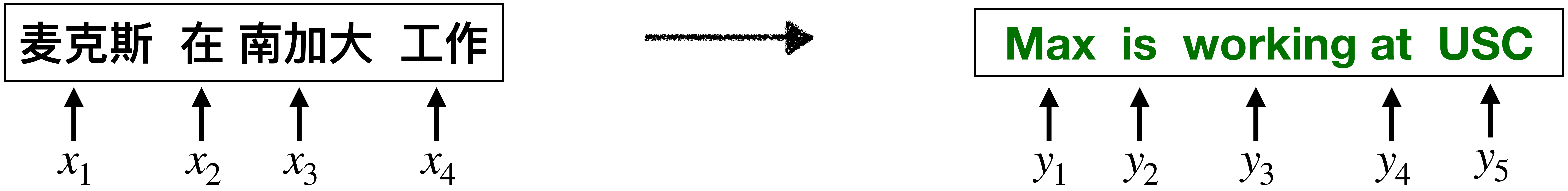
Seq2seq Generation

- **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



Statistic Machine Translation

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

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

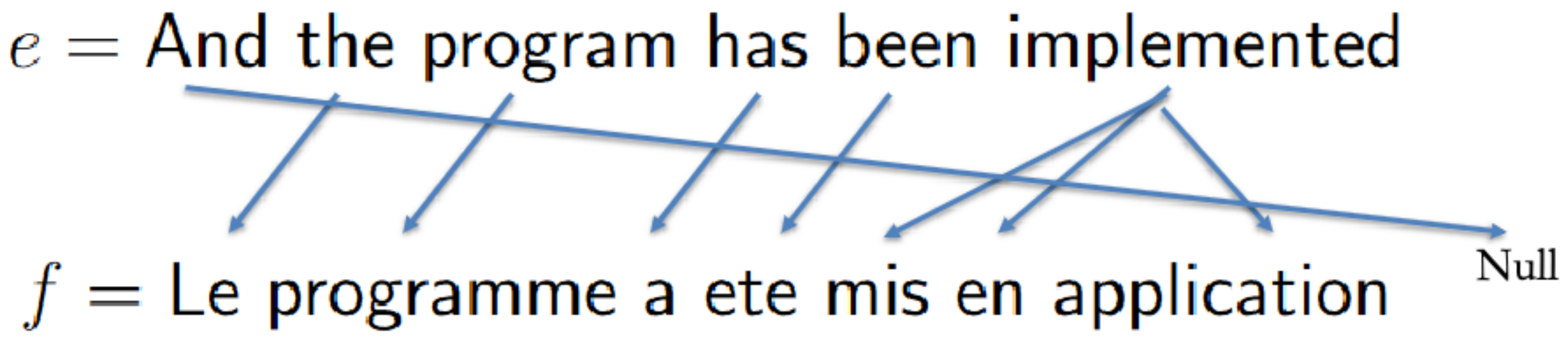
Word-Alignment Model in SMT

- **Key Idea:** two words are more likely to be aligned when they occur more frequently in translation pairs

我 不 知道	I	don’t	know	
我 是 学生	I	am	a	student
我 爱 喝 茶	I	love	drinking	tea

Word-Alignment Model in SMT

- e is an English sentence with l words
- f is a foreign sentence with m words
- An alignment $a = \{a_1, a_2, \ldots, a_m\}$, $a_j \in \{0, \ldots, l\}$
- Hence there are $(l+1)^m$ possible alignments



Word-Alignment Model in SMT

- **IBM Model 1:**

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

- **IBM Model 2:**

$$p(a \mid e, m) = q(a_j \mid j, l, m)$$

- **IBM Model 3, 4, 5, 6…**

1. This slide lists various IBM models for word alignment, starting from Model 1, which assumes uniform distribution for the alignments, meaning each word in the foreign sentence has an equal probability of being aligned with any word in the English sentence or the null alignment.
2. IBM Model 2 introduces a more refined probability model "q" that conditions the alignment probability on the positions of the words in both sentences, allowing for more accurate alignments that can consider positional biases.
3. Subsequent IBM models(3, 4, 5, 6...) introduce increasingly sophisticated mechanisms for modeling alignments, including fertility (how many times a word in the English sentence is used in the translation), reordering, and more.
4. The final equationover all possible alignments "a" expresses the probability of translating the foreign sentence "f" from the English sentence "e". This is done by summing over the probabilities of all possible alignments, weighted by the probability of the foreign sentence given a particular alignment and the English sentence.

Translation Model

$$p(f \mid e) = \sum_{a \in \mathcal{A}} p(a \mid e, m) p(f \mid a, e, m)$$

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: $p_\theta(Y|X)$ **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

麦克斯 在 南加大 工作 → **Max is working at USC**

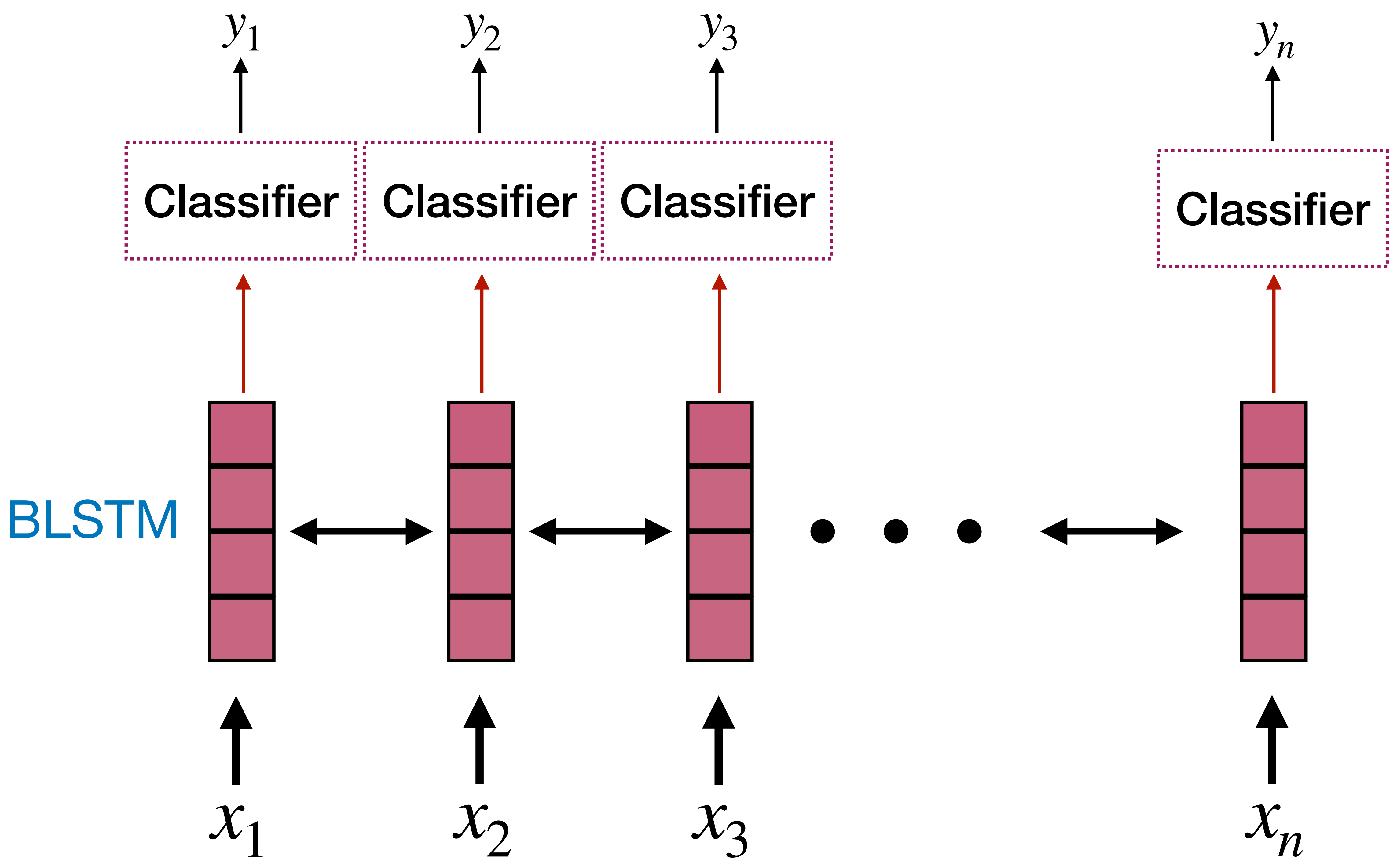
x_1 x_2 x_3 x_4 y_1 y_2 y_3 y_4 y_5

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

criticizes the naive use of sequence labeling for seq2seq generation tasks. It shows an incorrect approach where each output token y_t is generated independently given the entire input sequence X, without considering the previously generated tokens.

The examples show how the Chinese phrase "我 不知道" would translate to different English phrases ("I don't know", "I do not know", "I have no idea") without any context or coherence because each word is translated separately.

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

Not a good choice!

我 不 知道

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

Autoregressive Seq2seq Generation

- **Autoregressive Factorization:**

Here, each output token is predicted one after another, with each prediction taking into account not only the input sequence but also the previously predicted tokens.

This way, the model can use the context from what it has already generated to inform the next token, creating a coherent and contextually appropriate sequence.

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

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

麦克斯	在	南加大	工作
x_1	x_2	x_3	x_4

$\longrightarrow$

Max	is	working	at	USC
y_1	y_2	y_3	y_4	y_5

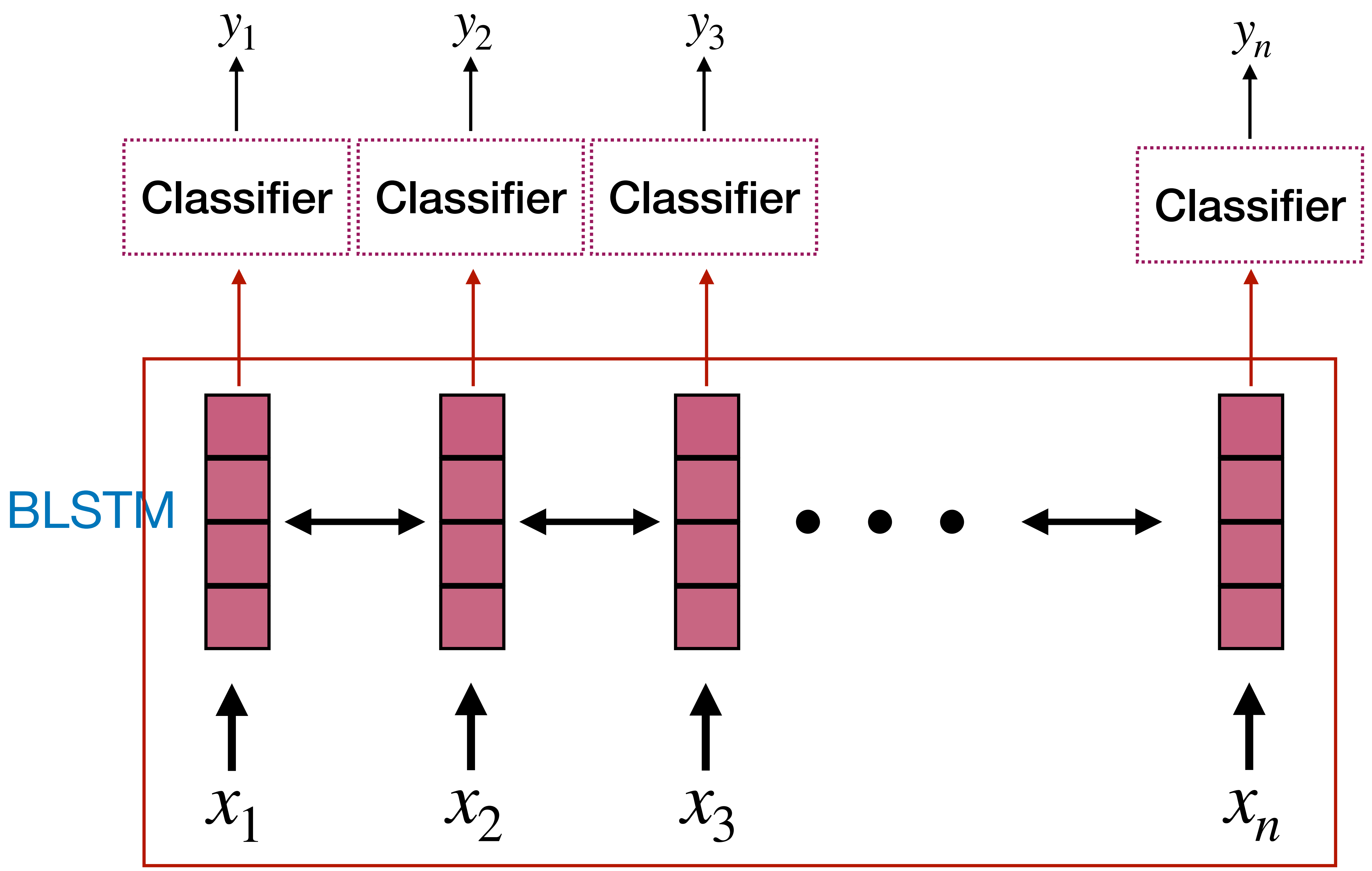
Encoder-Decoder Architecture

- **Sequence labeling vs. Seq2seq Generation**

Sequence labeling is like assigning a category or tag to each item in a sequence independently. Imagine you have a row of boxes, and for each box, you independently decide what type of object should go inside based on the label on the box.

Sequence labeling

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



Seq2seq Generation

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

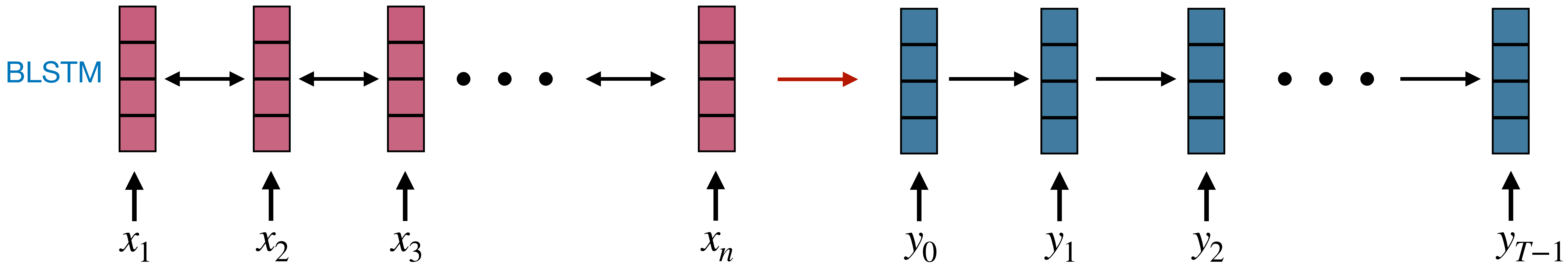
Seq2seq generation is more complex. It's like telling a story where each word you say depends on the words you've already said. So the "story" (output sequence) unfolds one word at a time, and each new word takes into account the whole story so far.

Encoder: encode a sentence into a sequence of vectors

Decoder: use another LSTM?

Encoder-Decoder Architecture

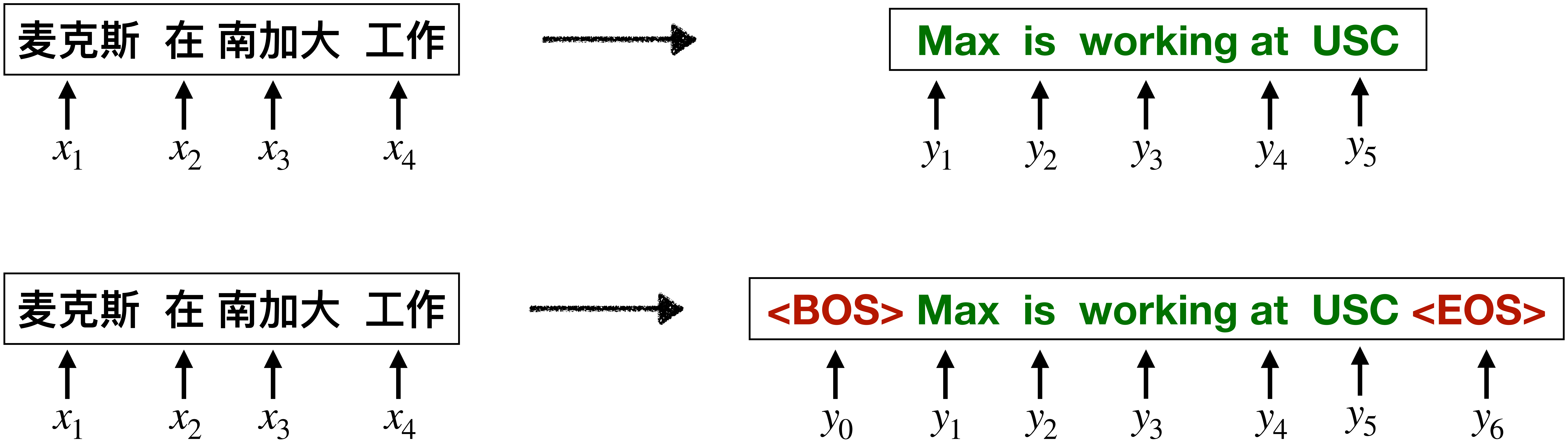
- **Two Components:**
  - Encoder: Convert input sequence into a sequence of vectors
  - Decoder: Convert encoding into a sequence in the output space



Special Tokens in Seq2seq

- <BOS>: start of the target sentence
- <EOS>: end of the target sentence

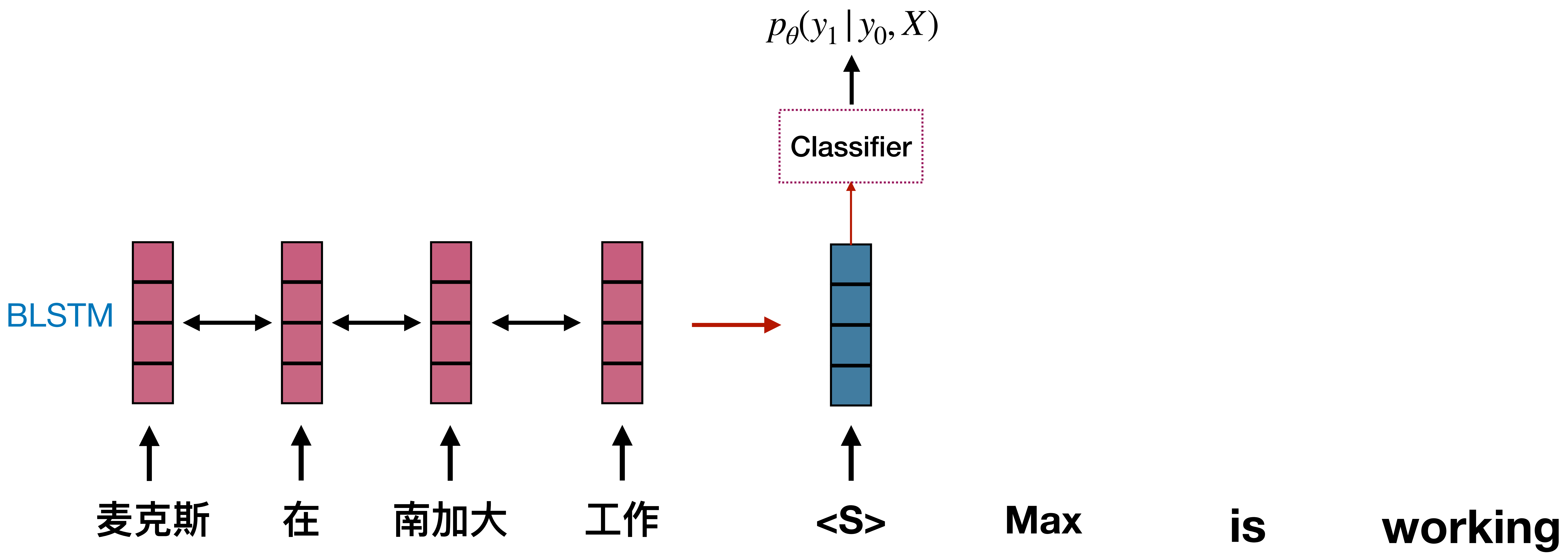
These tokens are important because they provide clear markers for the beginning and end of a sentence, which helps the model in both training and inference phases to know when to start and stop generating text.



Seq2seq Training

- **Model Training:**

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

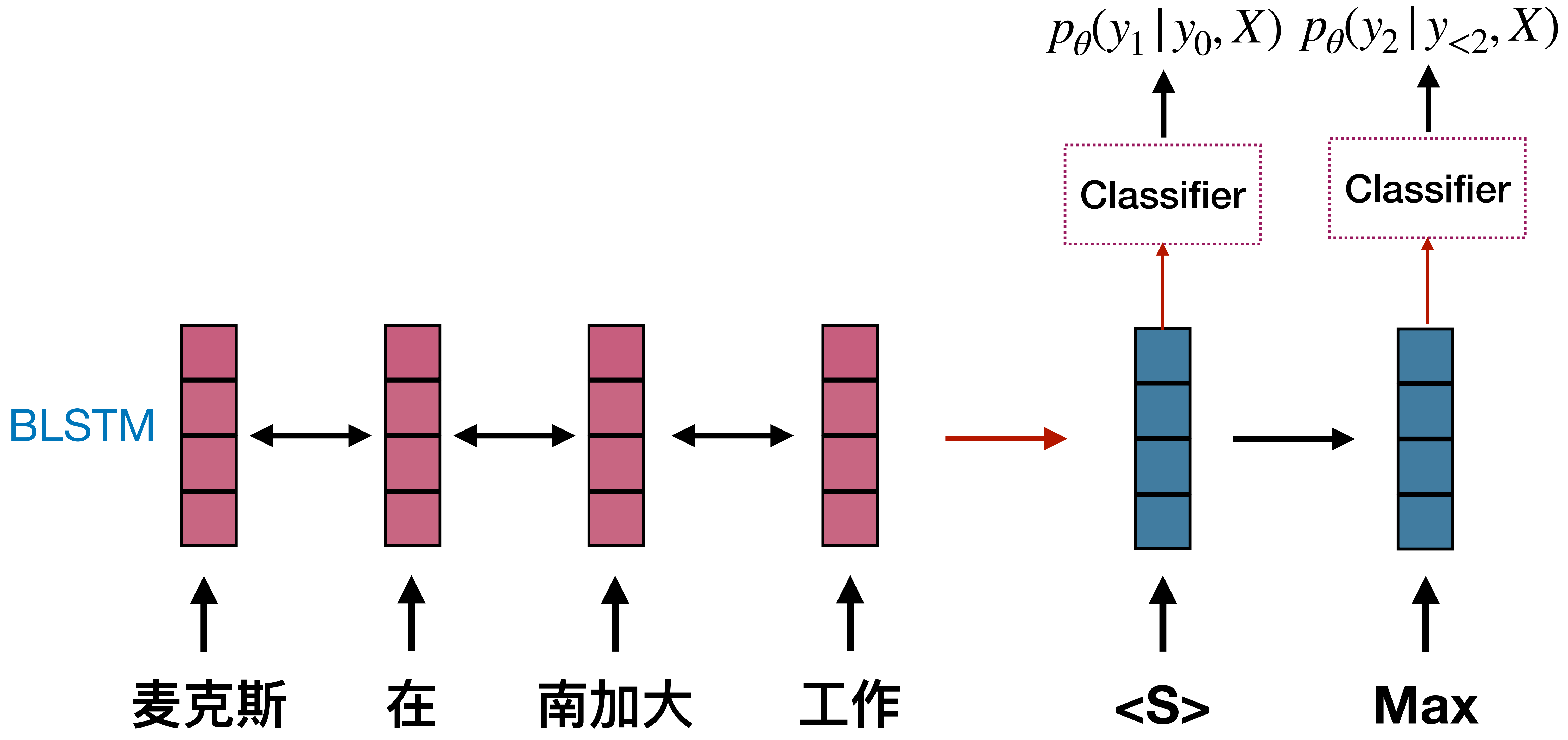


Seq2seq Training

- **Model Training:**

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

Each word in the output is predicted based on the entire input sentence and all the previously predicted words in the output sentence. This is shown by the "Classifiers" in the slides, which represent the decision-making at each step for predicting the next word.

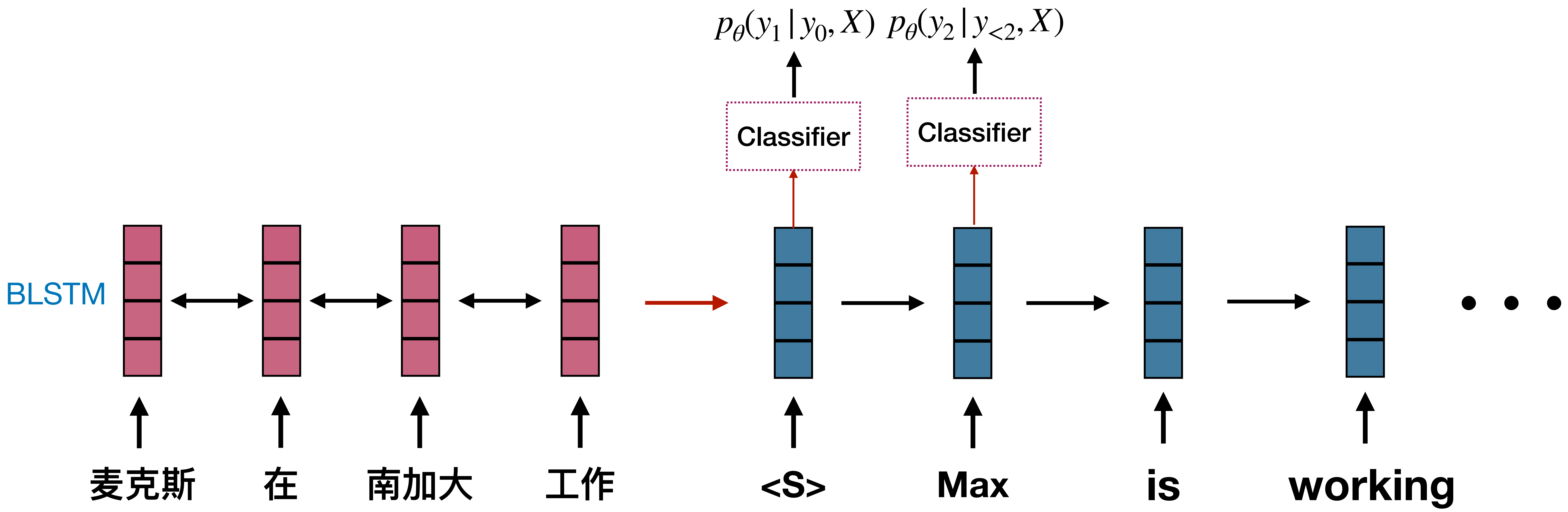


is **working**

Seq2seq Training

- **Model Training:**

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



Seq2seq Training

- **Maximum Likelihood Estimation**

The goal of training is to adjust the model parameters (denoted by θ) to maximize the probability of the correct output sequence given the input sequence. This is called 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

the model's predictions are compared to the actual correct translation, and then the model adjusts its parameters to reduce the difference.

36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

Seq2seq Decoding

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

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

exhaustive search would look at all possible sequences that can be generated and choose the one with the highest probability. This is computationally very expensive.

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

Instead of looking at all possible sequences, the model just picks the most likely next word at each step.

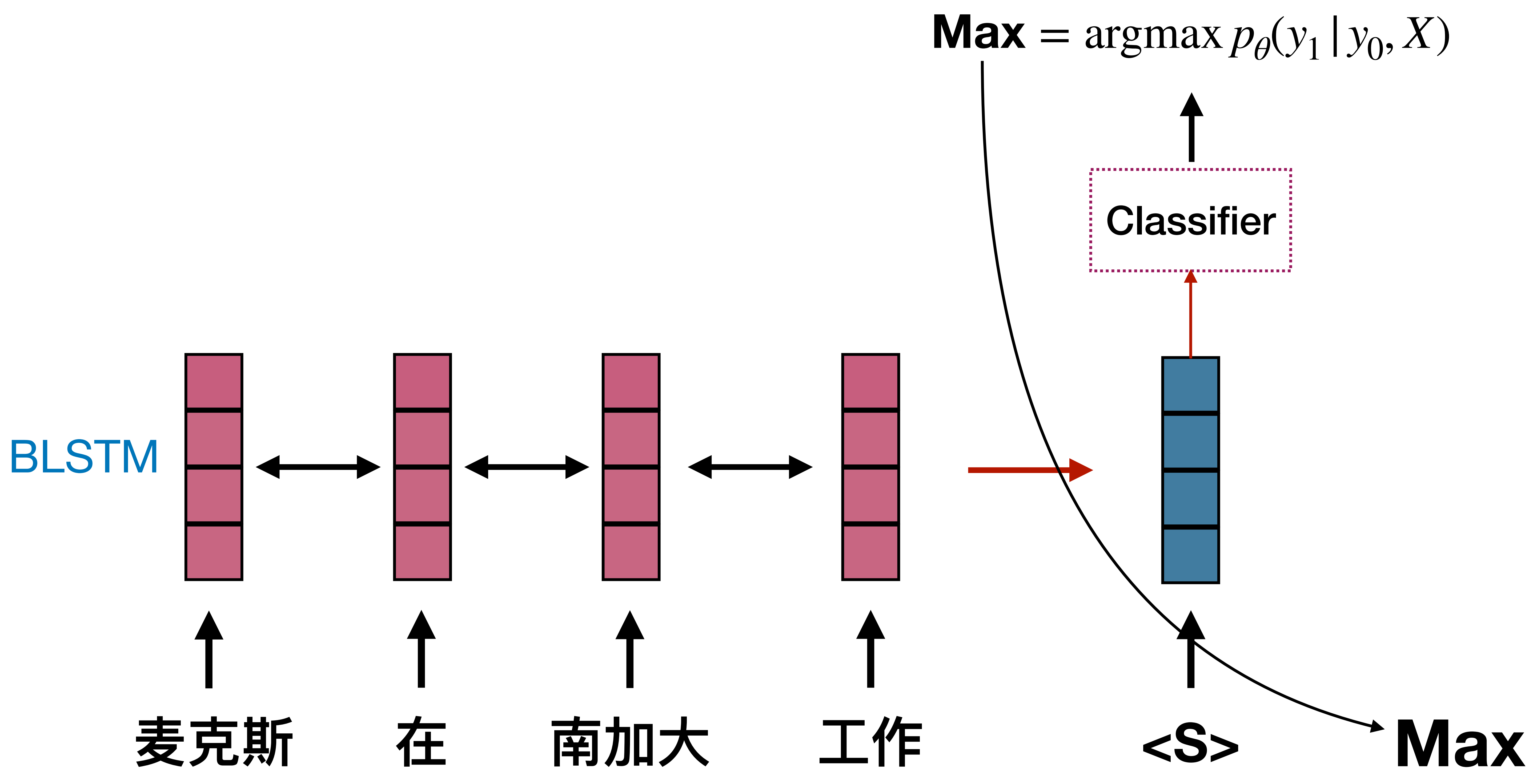
$$\arg\max_{Y} p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t|y_{<t}, X)$$

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

Seq2seq Decoding

- **Greedy decoding:**

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

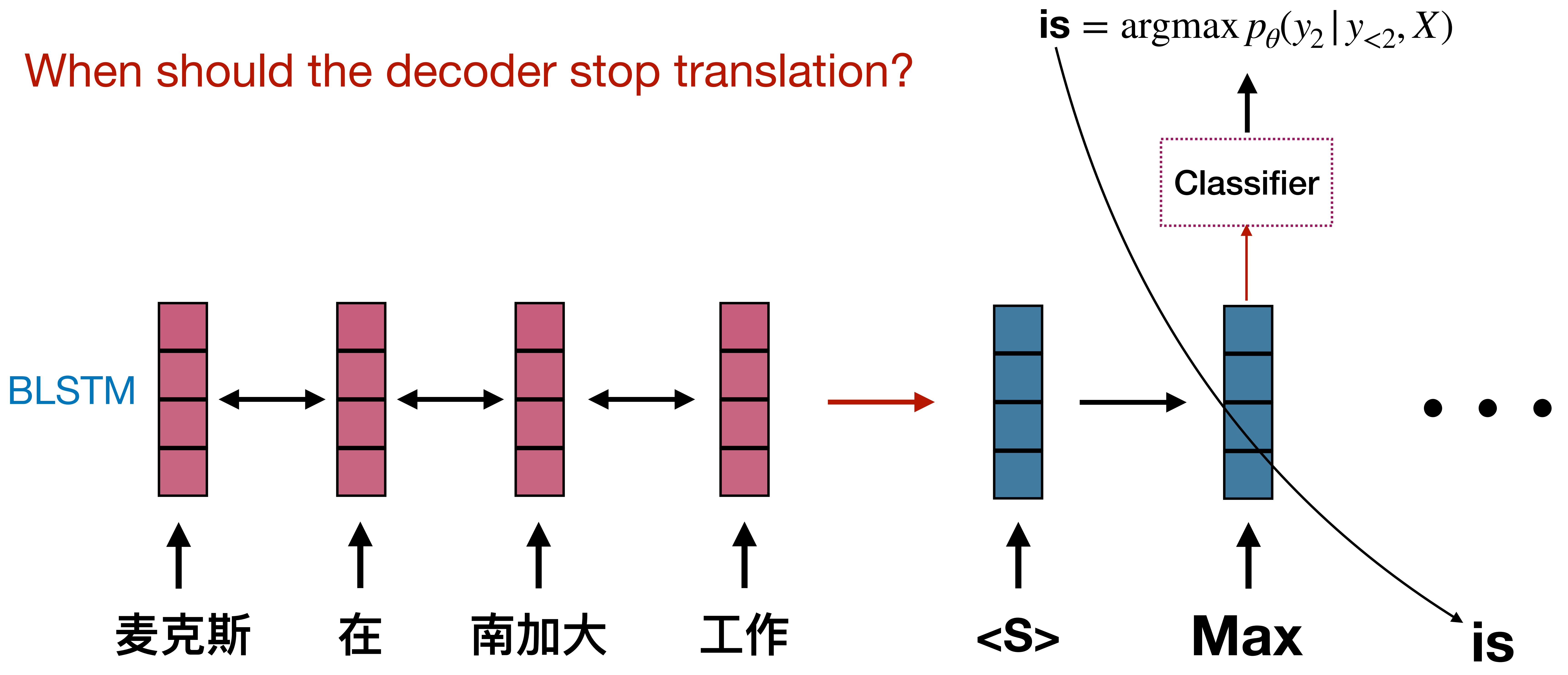


Seq2seq Decoding

- **Greedy decoding:**

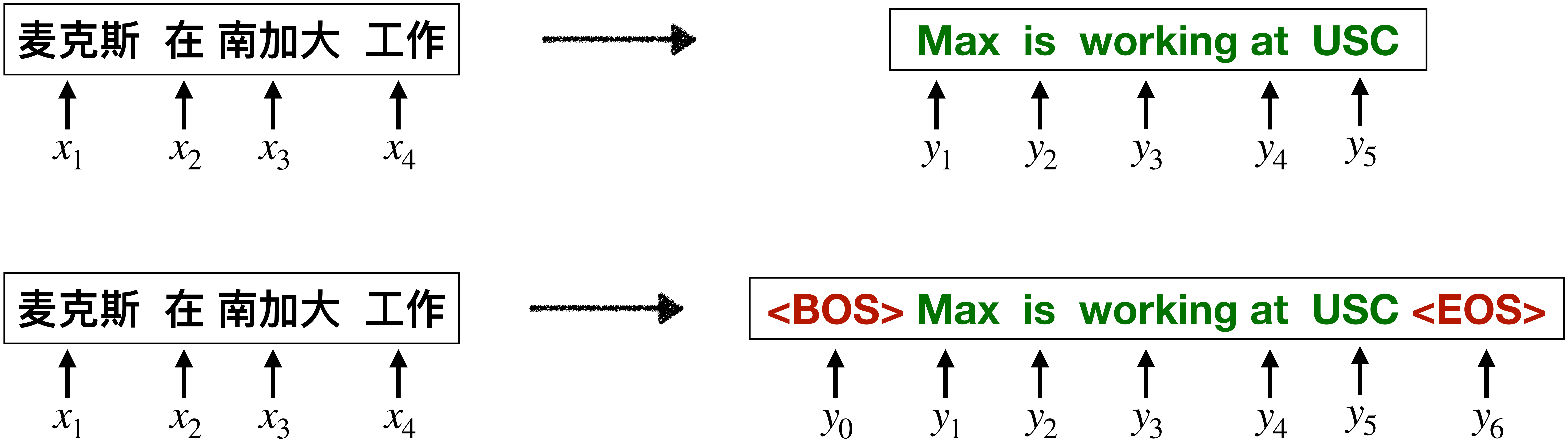
$$y_t^* = \arg\max_{y_t} p_\theta(y_t \mid y_{<t}, X), \forall t$$

When should the decoder stop translation?



Special Tokens in Seq2seq

- <BOS>: start of the target sentence
- <EOS>: end of the target sentence



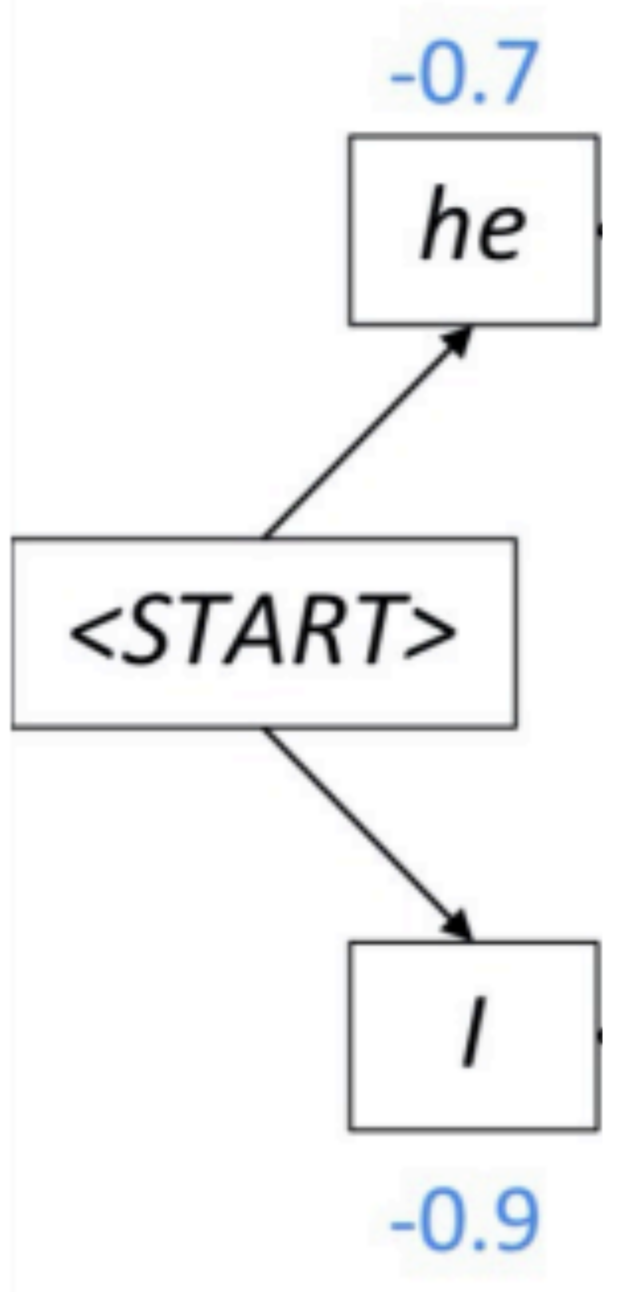
A Middle Ground: Beam Search

- **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 is an optimized search strategy that, instead of considering all possible translations at each step of the decision process, narrows down the options to a manageable number by keeping track of only the K most probable partial translations (also known as hypotheses). This approach strikes a balance between efficiency and the likelihood of finding a high-quality solution.

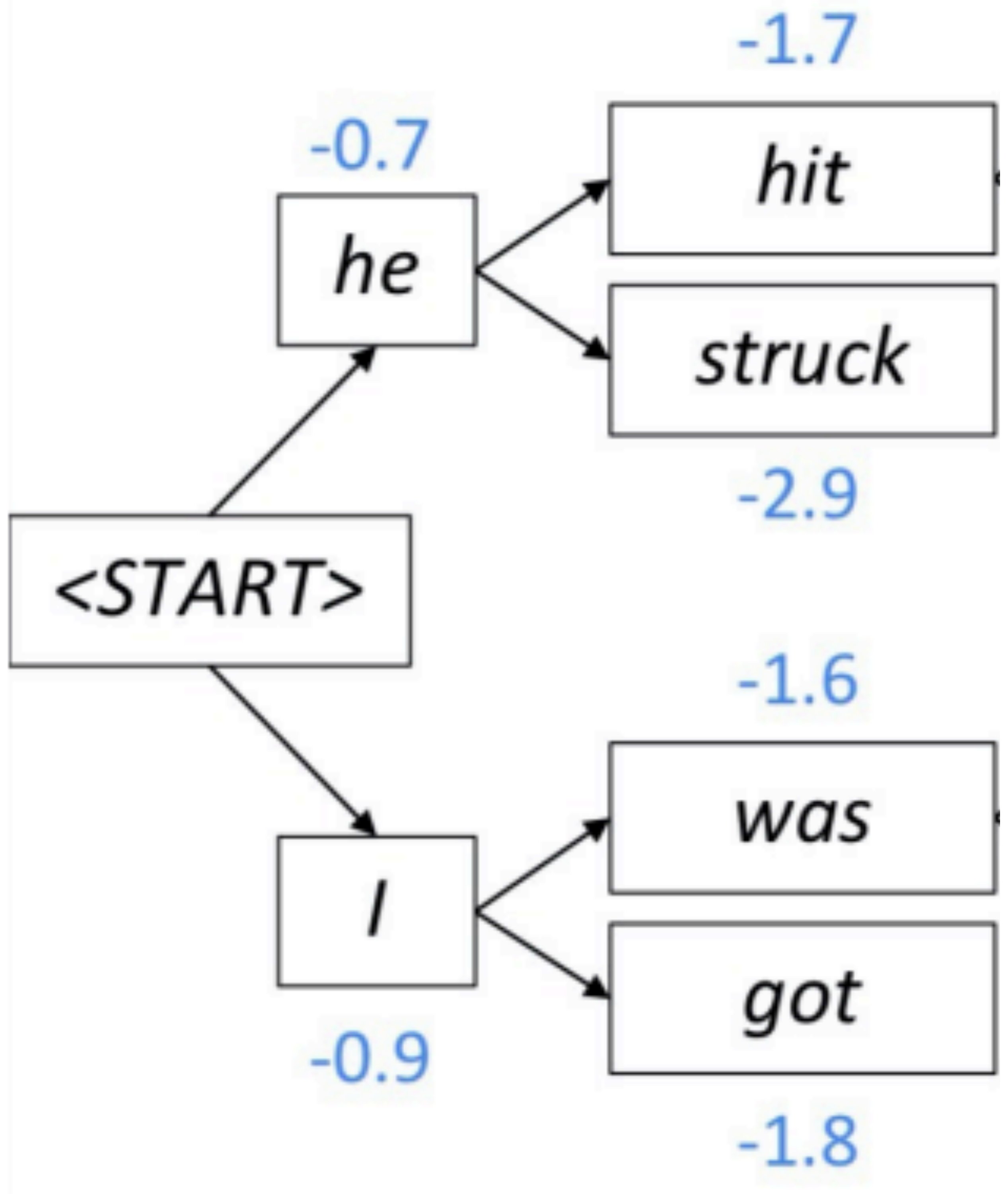
Beam Search Decoding

Beam size $K = 2$



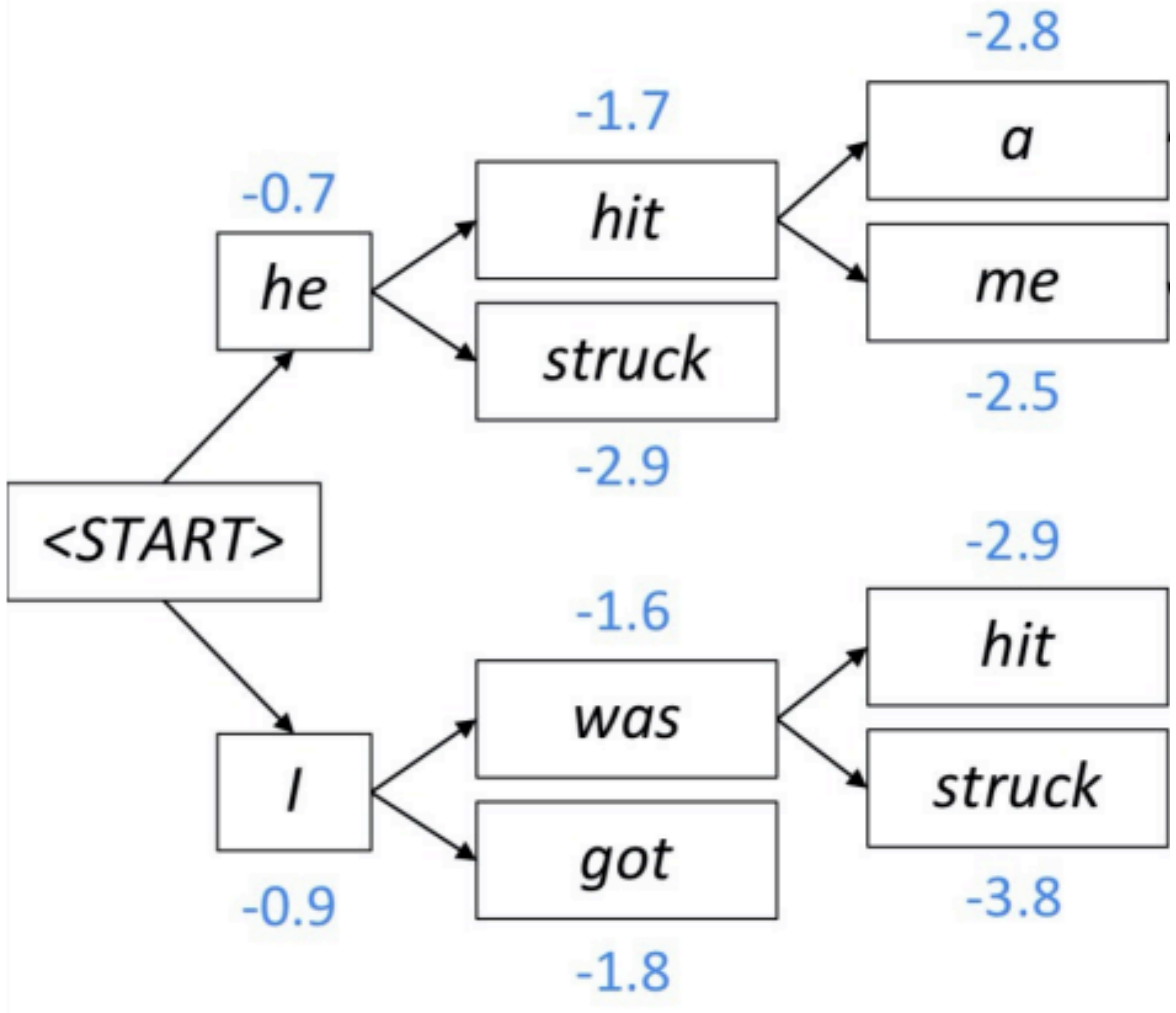
Beam Search Decoding

Beam size $K = 2$



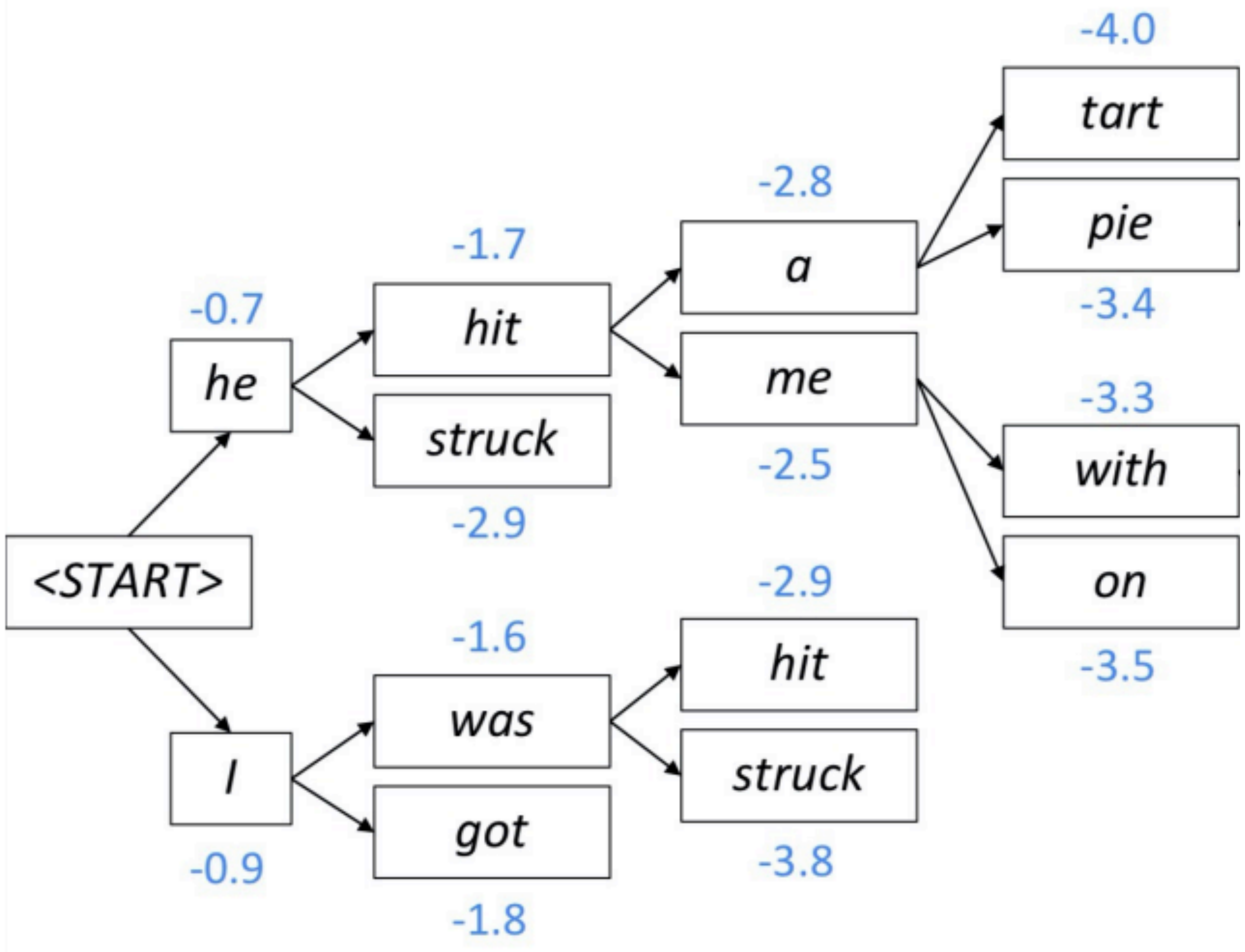
Beam Search Decoding

Beam size $K = 2$



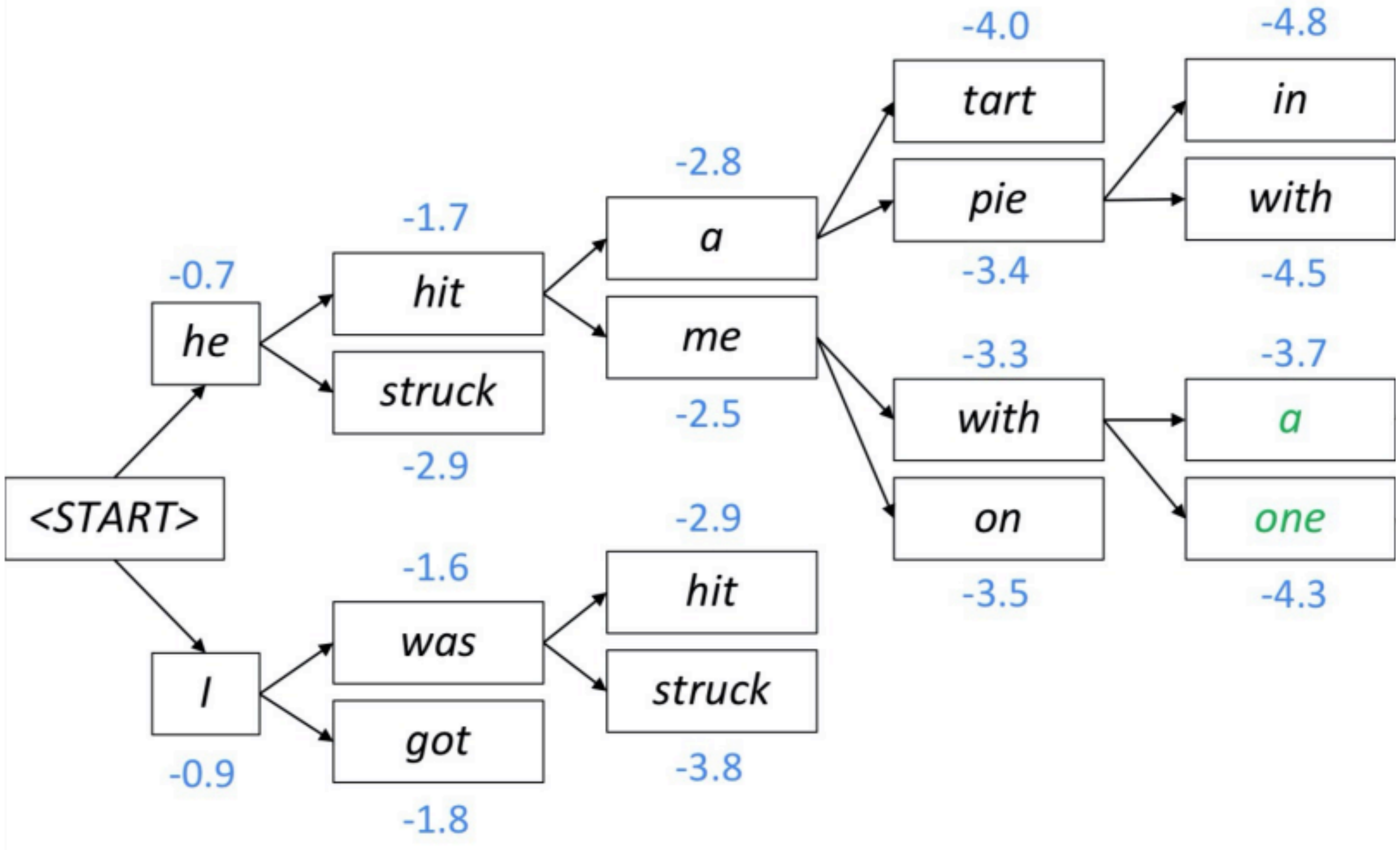
Beam Search Decoding

Beam size $K = 2$



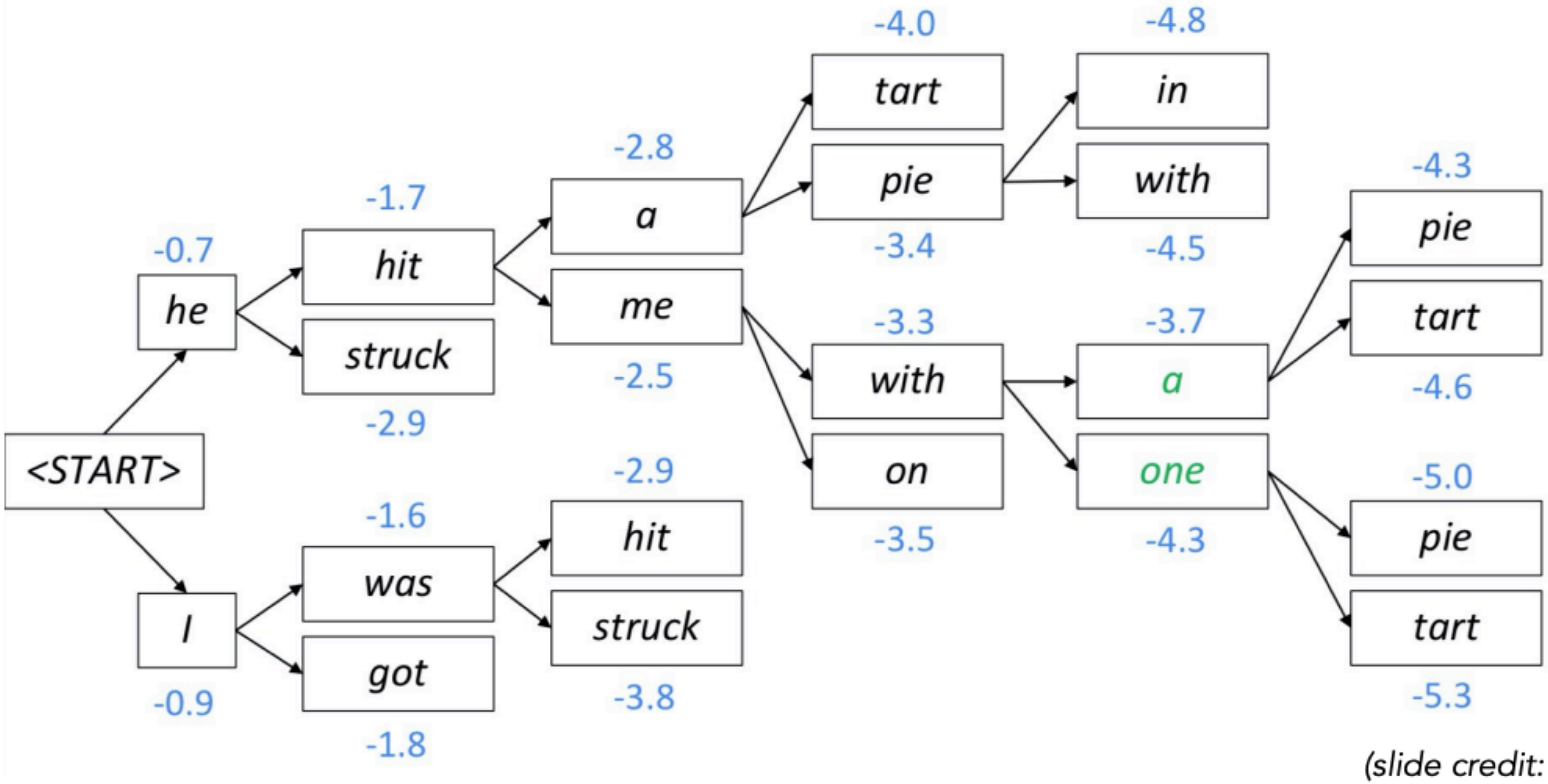
Beam Search Decoding

Beam size $K = 2$



Beam Search Decoding

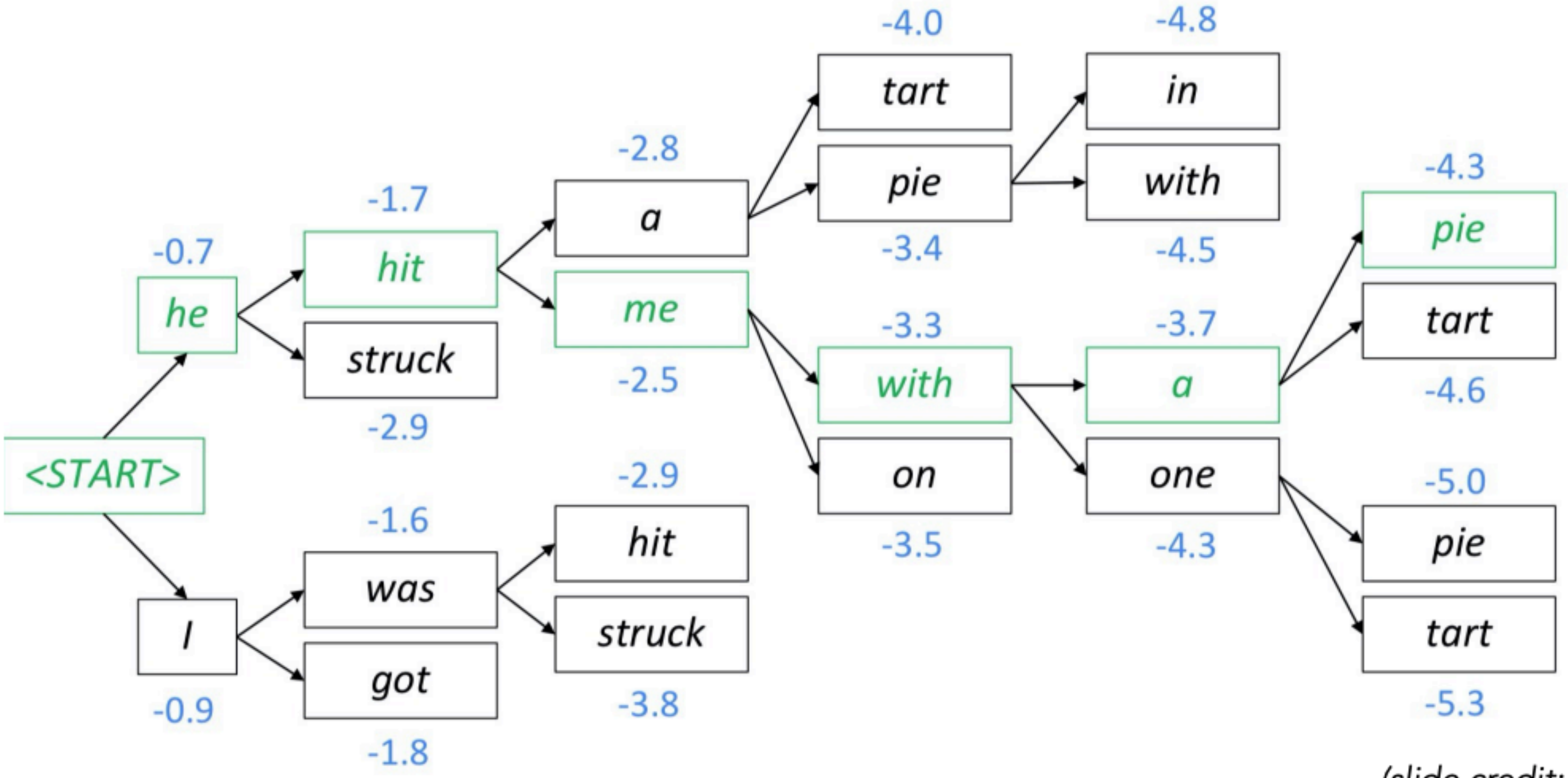
Beam size $K = 2$



(slide credit: Abigail See)

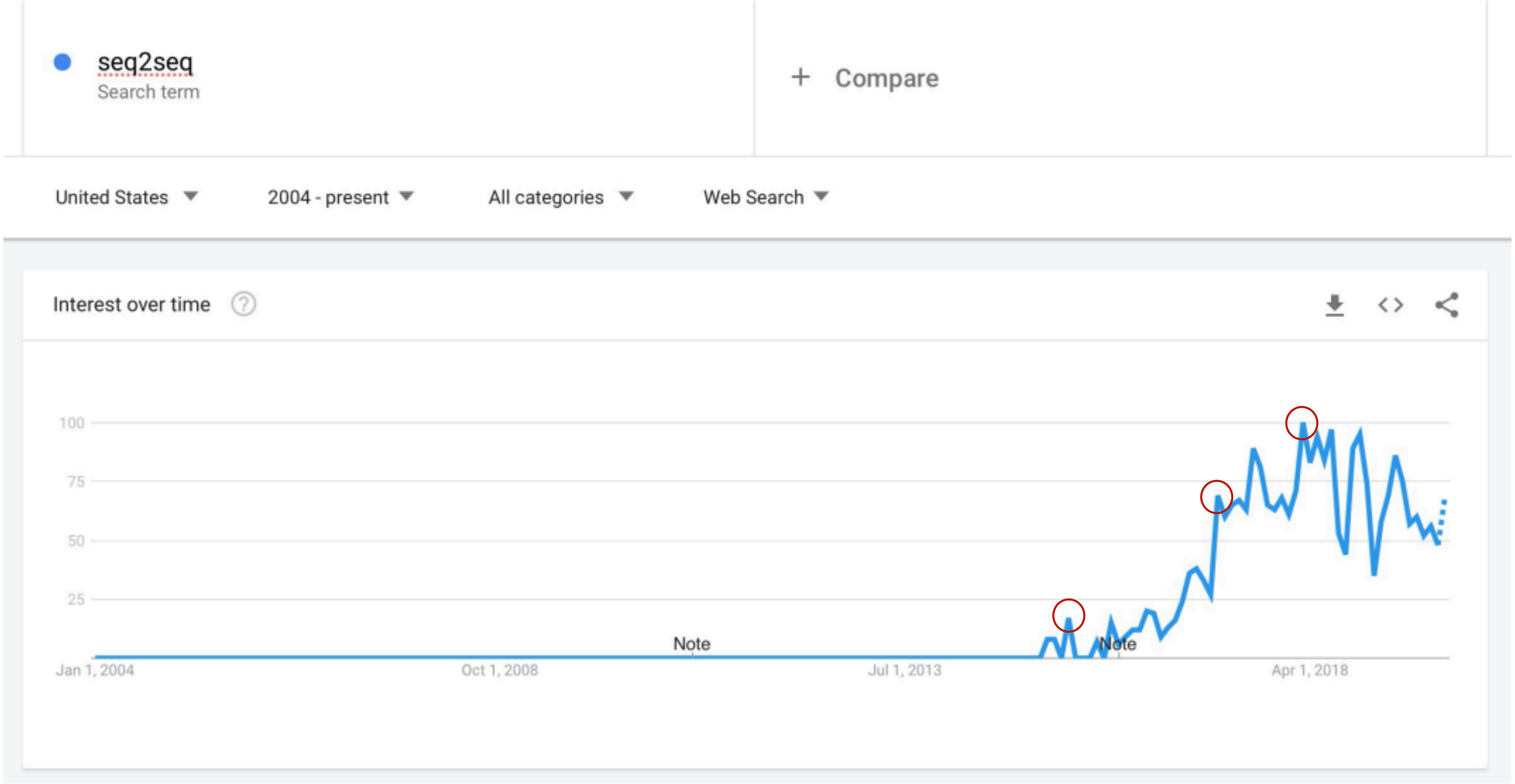
Backtrack

Beam size $K = 2$

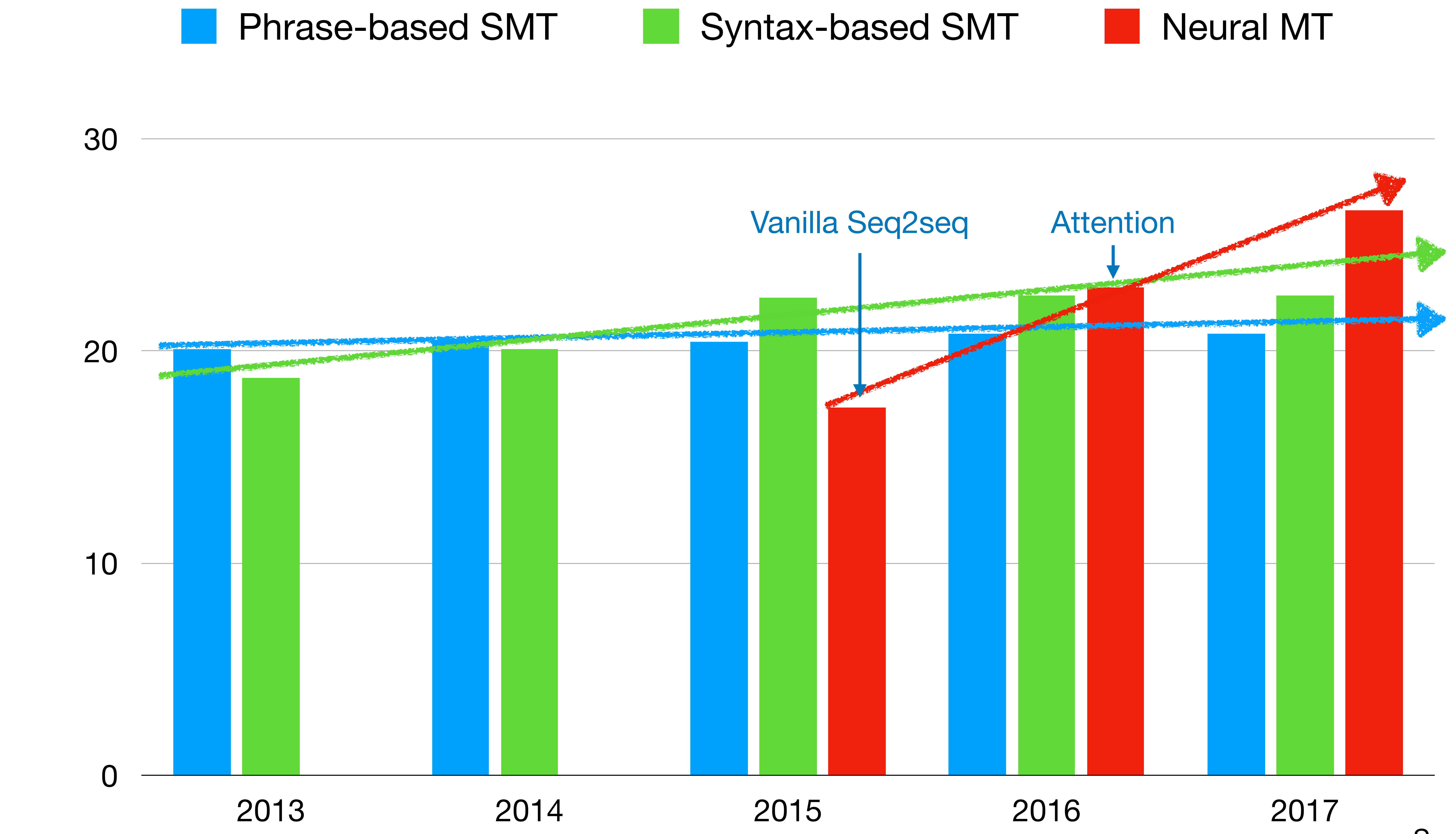


(slide credit: Abigail See)

How Seq2seq changed the MT Landscape



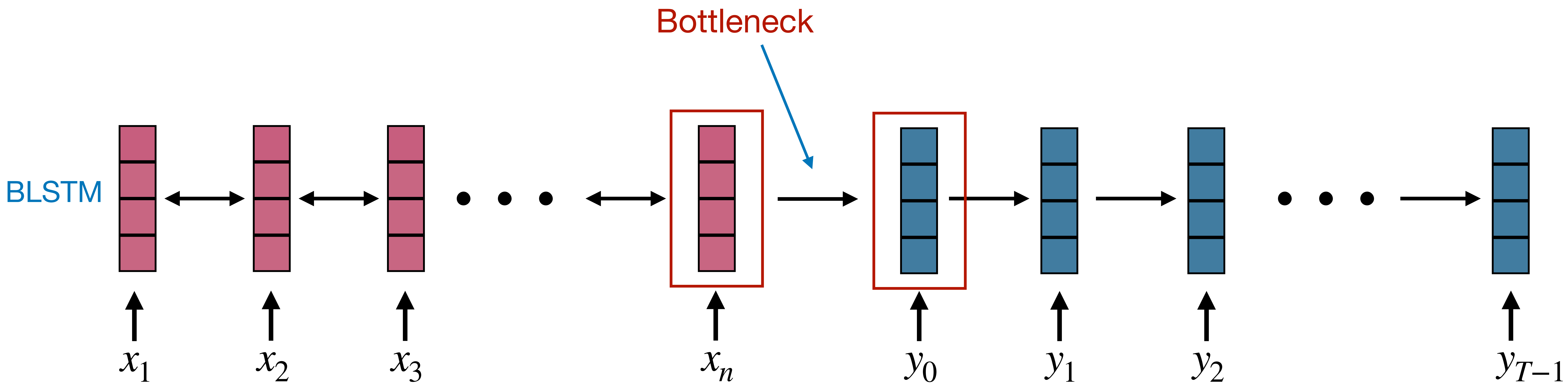
MT Progress



Source: Rico Sennrich

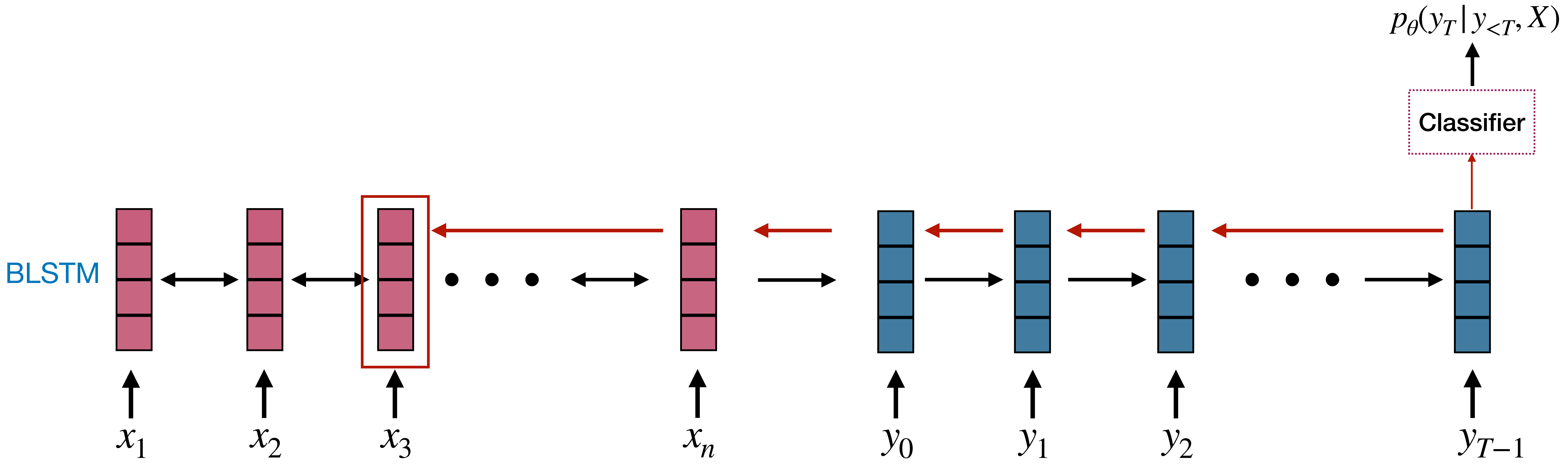
Issues with Vanilla Encoder-Decoder Architecture

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



Issues with Vanilla Encoder-Decoder Architecture

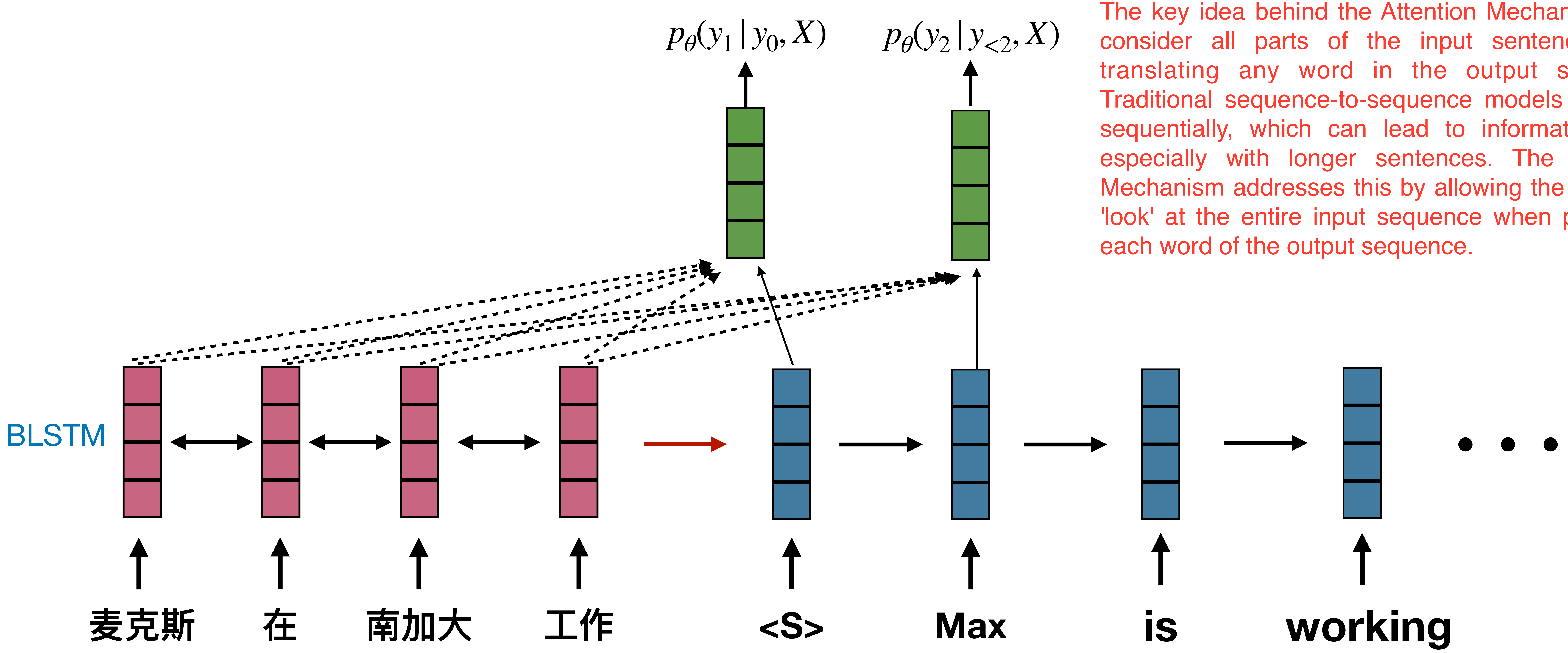
- A single encoding vector needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients



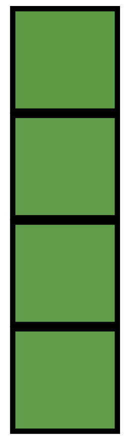
Attention Mechanism

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

The key idea behind the Attention Mechanism is to consider all parts of the input sentence when translating any word in the output sentence. Traditional sequence-to-sequence models translate sequentially, which can lead to information loss, especially with longer sentences. The Attention Mechanism addresses this by allowing the model to 'look' at the entire input sequence when predicting each word of the output sequence.

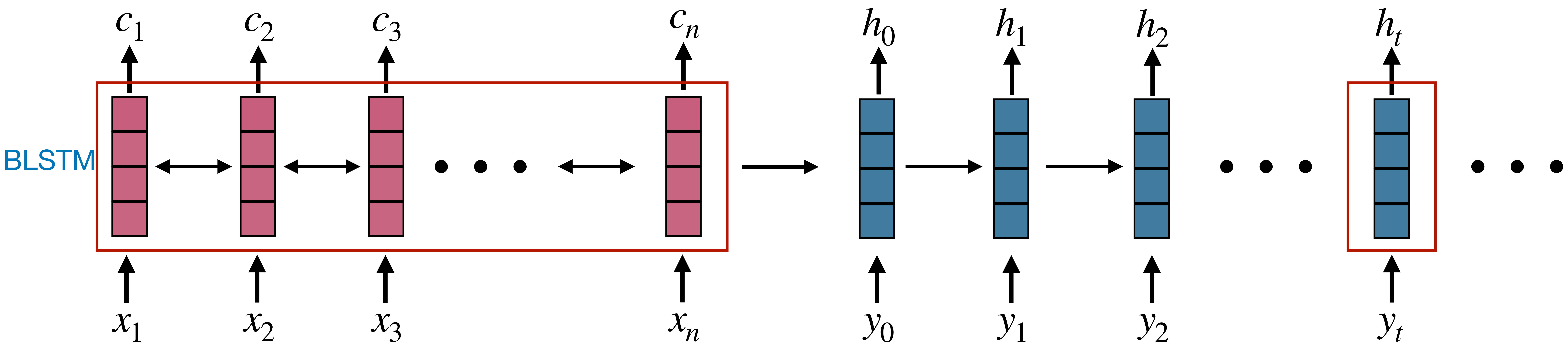


Attention Mechanism

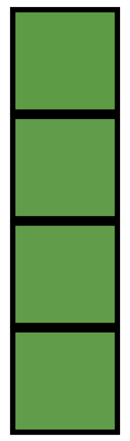


$$= \text{attn}([c_1, c_2, \ldots, c_n], h_t)$$

In essence, for every output word being predicted, the mechanism computes a set of attention scores. Each score represents the relevance of an input word to the output word currently being predicted.



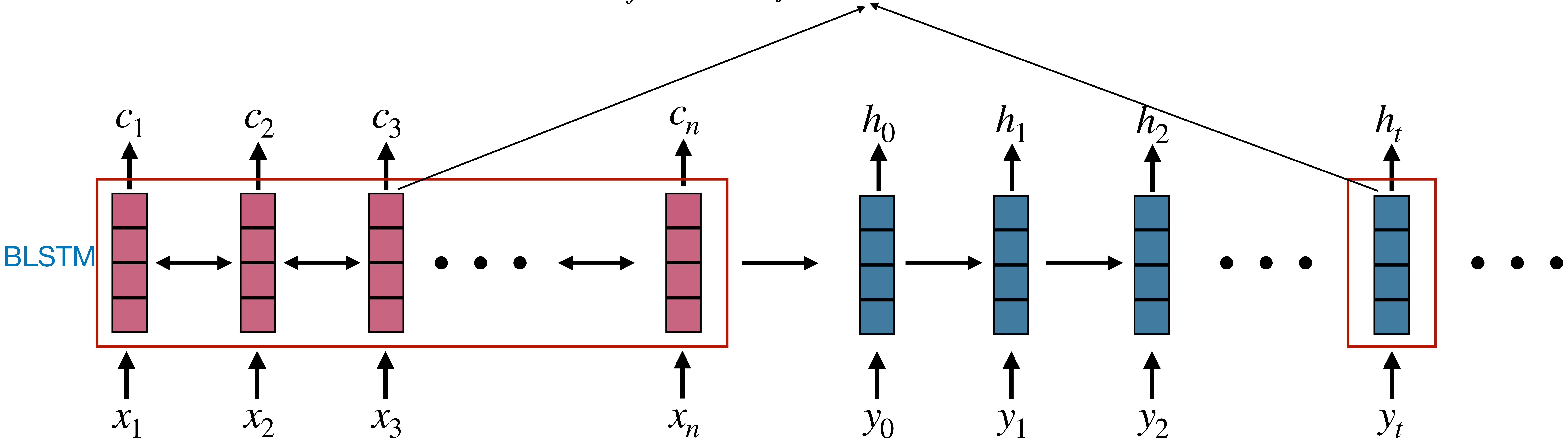
Attention Mechanism



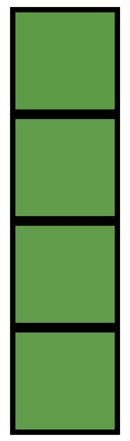
$$= \mathrm{attn}([c_1, c_2, \ldots, c_n], h_t)$$

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

Attention scores



Attention Mechanism

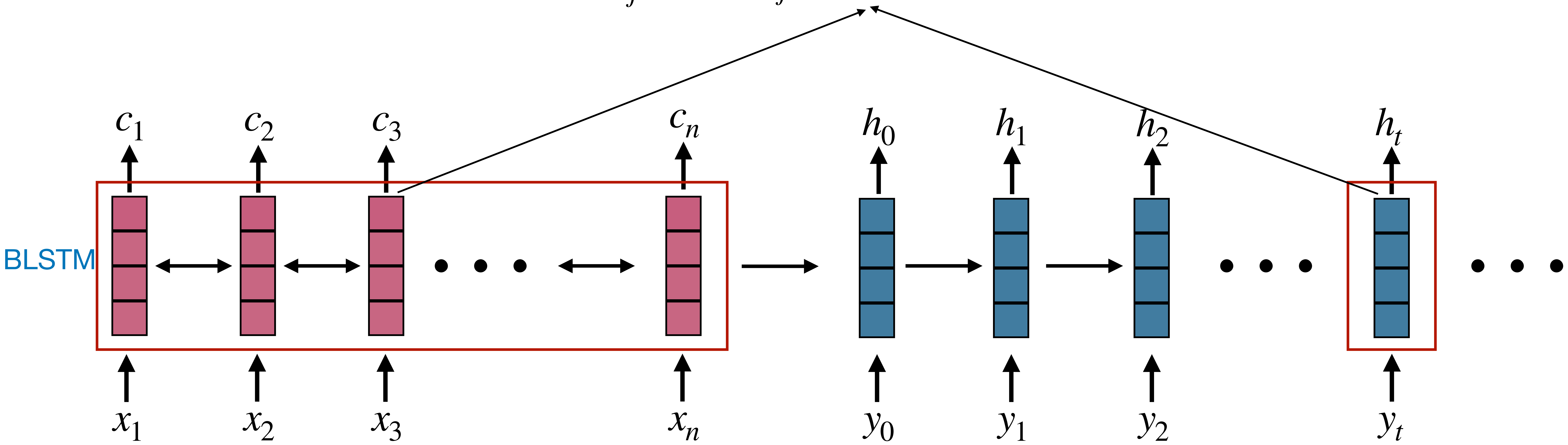
 $= \text{attn}([c_1, c_2, \ldots, c_n], h_t)$

$$a^t = \text{softmax}(e^t) \in (0,1)^n$$

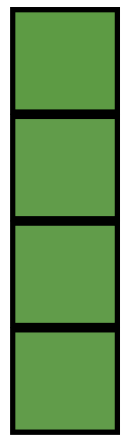
Attention distribution

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

Attention scores



Attention Mechanism



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

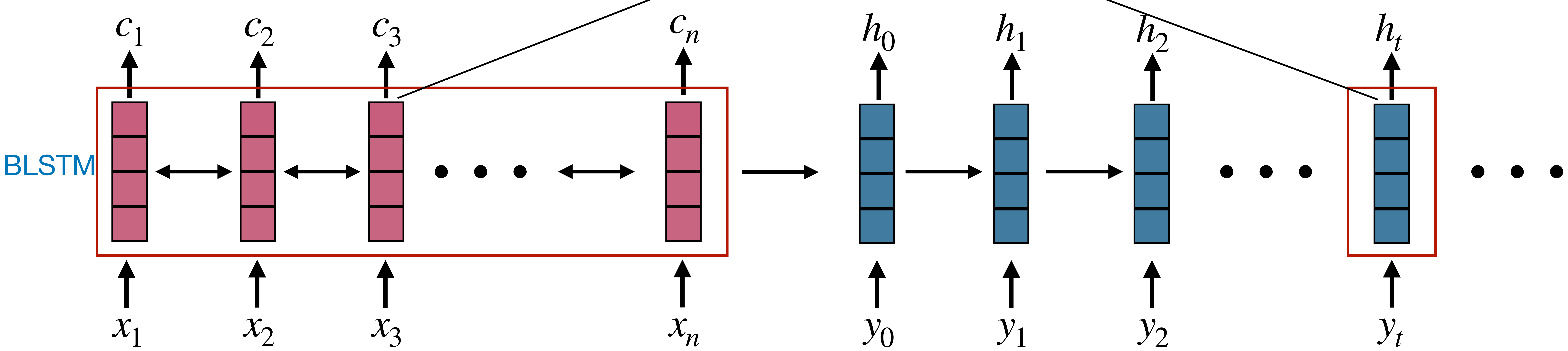
Attention output

$$a^t = \mathrm{softmax}(e^t) \in (0,1)^n$$

Attention distribution

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

Attention scores



Softmax Function

$$e^t = [e_1^t, e_2^t, \ldots, e_n^t]$$

$$\text{softmax}(e^t) = [\frac{\exp(e_1^t)}{\sum_{j=1}^{n} \exp(e_j^t)}, \frac{\exp(e_2^t)}{\sum_{j=1}^{n} \exp(e_j^t)}, \ldots, \frac{\exp(e_n^t)}{\sum_{j=1}^{n} \exp(e_j^t)}]$$

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 ›

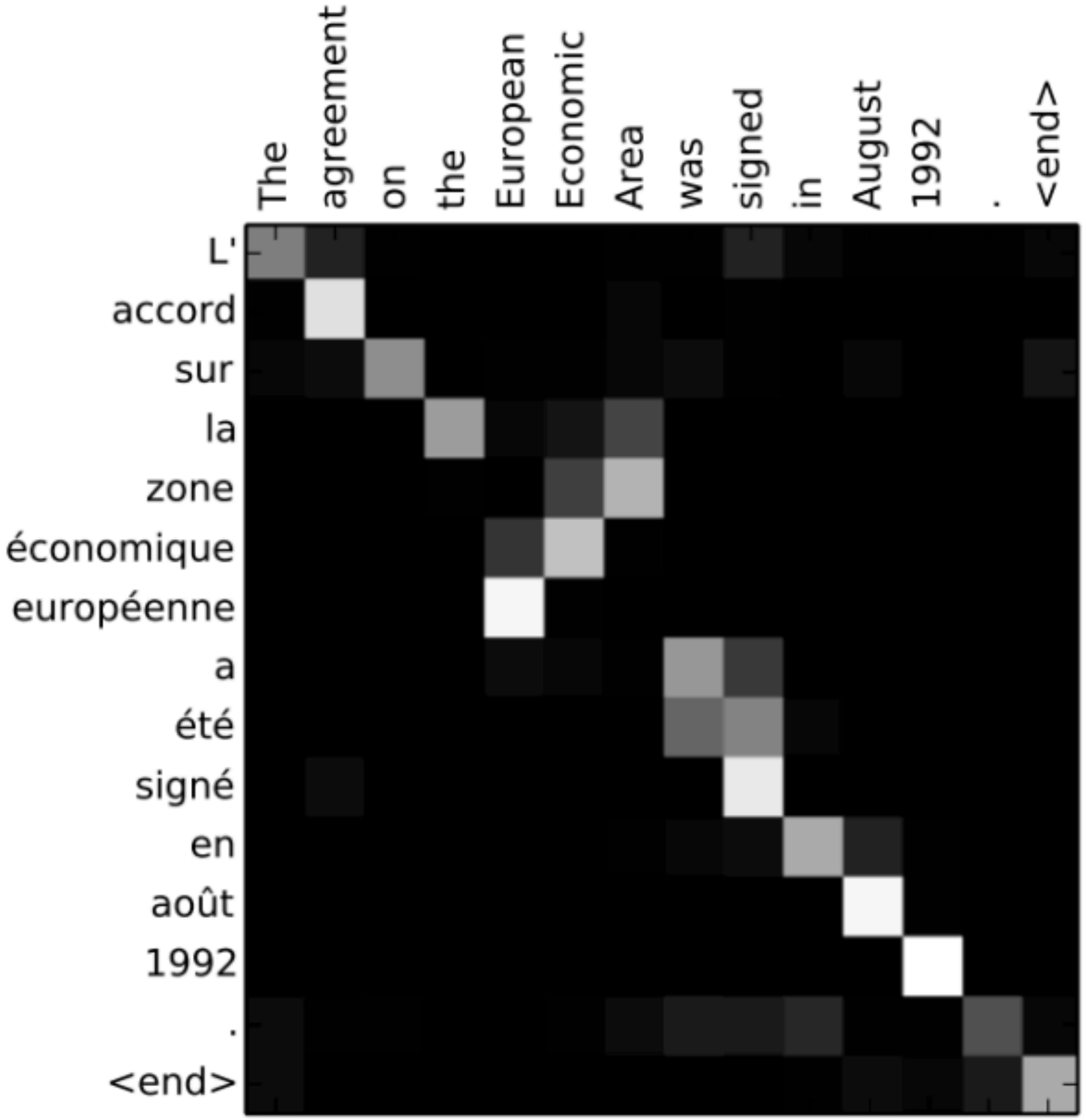
Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

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

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

(Wu et al., 2016)

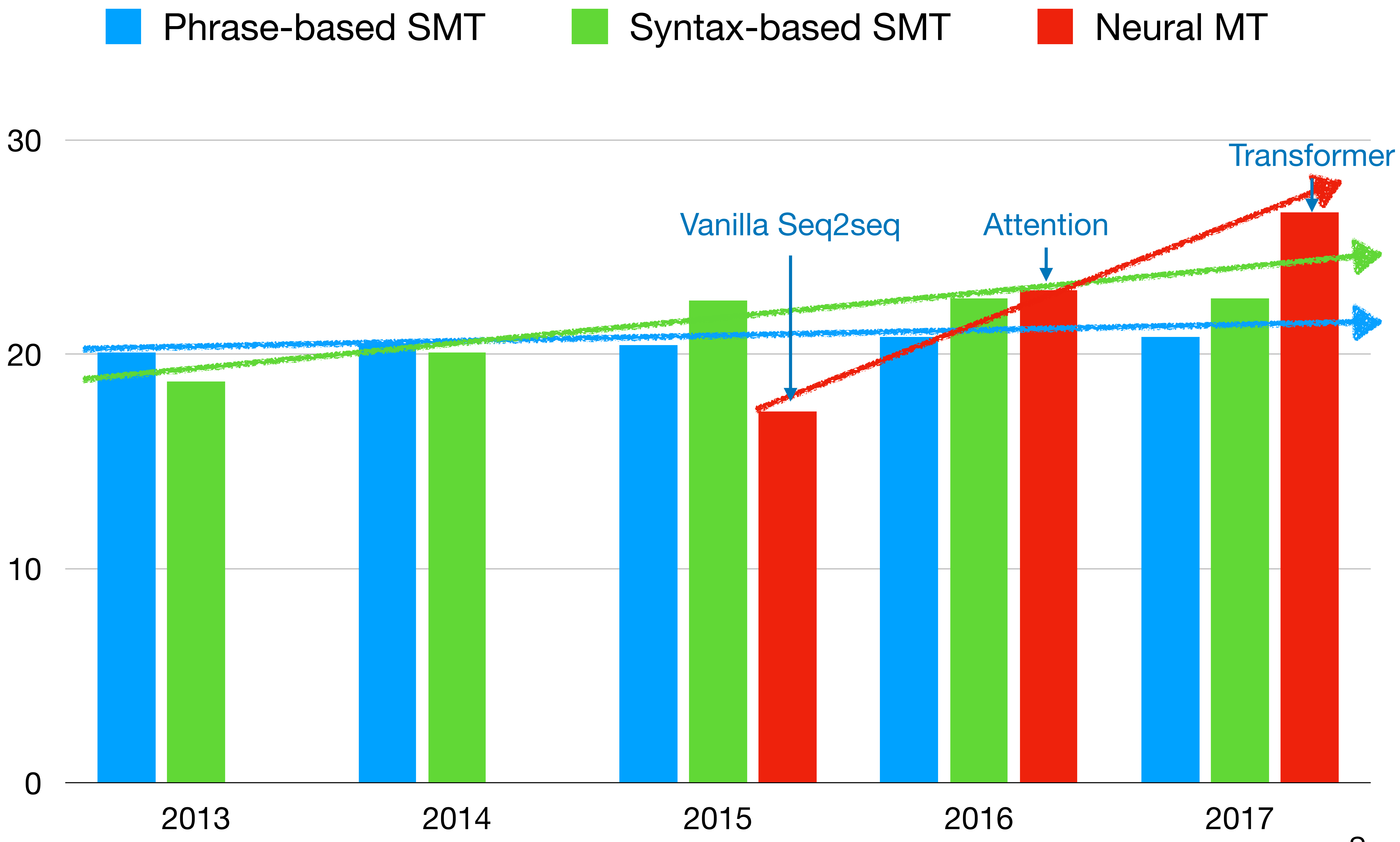
Visualizing Attention



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