Machine Translation, Encoder-Decoder Models and Attention

Alan Ritter

(many slides from Greg Durrett)

This Lecture

- Machine Translation Basics
- Seq2Seq / Encoder-Decoder Models
- Attention
- Decoding Strategies
- Transformers

MT Basics

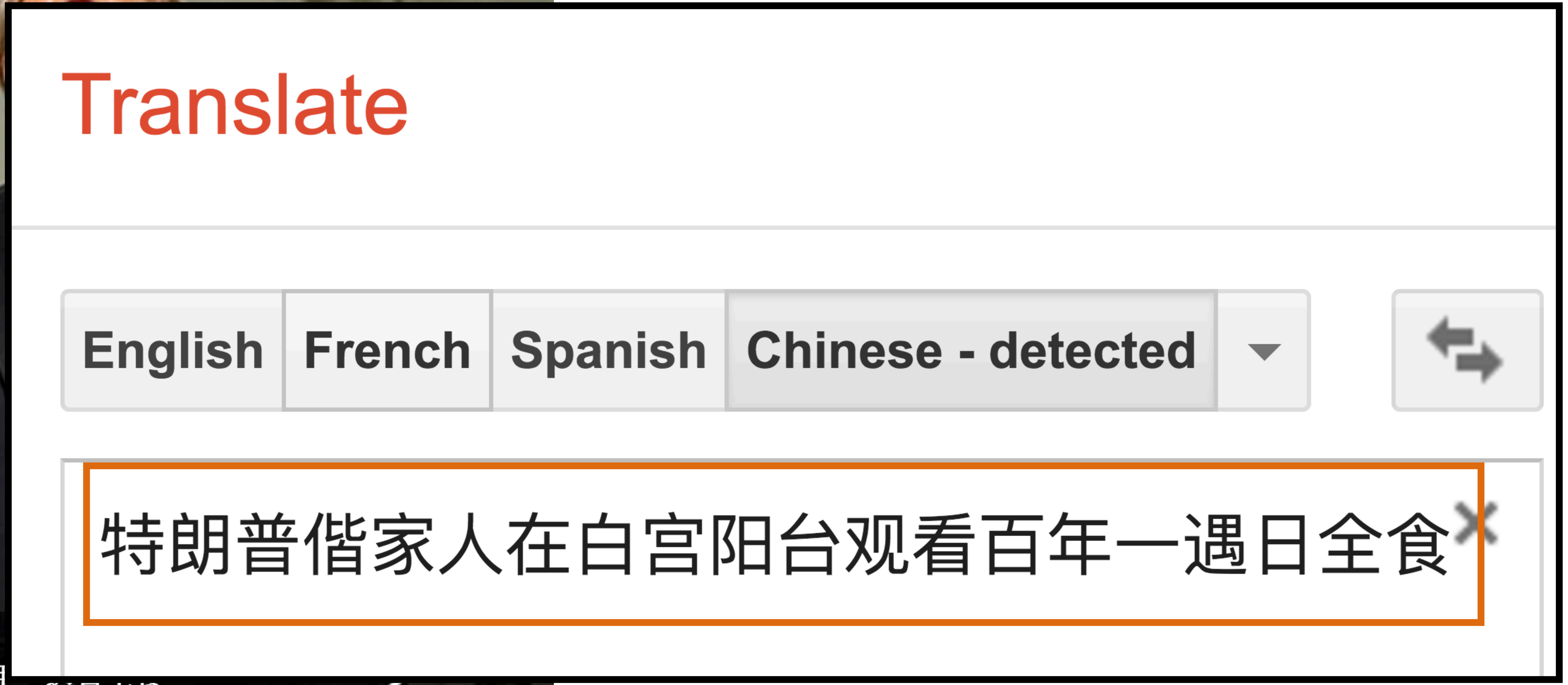
MT Basics

People’s Daily, August 30, 2017

MT Basics

People’s Daily, August 30, 2017

MT Basics



People's Daily, August 30, 2017

Trump Pope family watch a hundred years a year in the White House balcony

MT Basics

People's Daily, August 30, 2017

Translate

English | French | Spanish | Chinese - detected

特朗普偕家人在白宫阳台观看百年一遇日全食

Trump Pope family watch a hundred years a year in the White House balcony

Phrase-Based MT

- Key idea: translation works better the bigger chunks you use

Phrase-Based MT

- Key idea: translation works better the bigger chunks you use

- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate

Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext

Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language

Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language
  - Decoder takes phrases and a language model and searches over possible translations

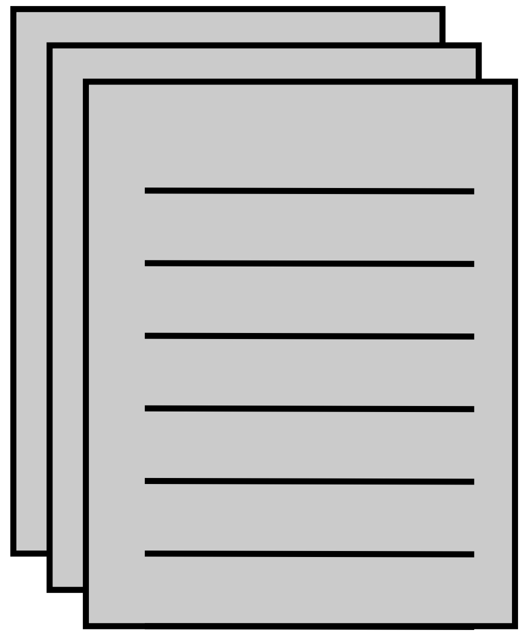
Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language
  - Decoder takes phrases and a language model and searches over possible translations
- NOT like standard discriminative models (take a bunch of translation pairs, learn a ton of parameters in an end-to-end way)

Phrase-Based MT

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Language model P(e)

Unlabeled English data

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

“Translate faithfully but make fluent English”

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

		1-gram	2-gram	3-gram
hypothesis 1	I am exhausted	3/3	1/2	0/1
hypothesis 2	Tired is I	1/3	0/2	0/1
hypothesis 3	I I I	1/3	0/2	0/1
reference 1	I am tired			
reference 2	I am ready to sleep now and so exhausted			

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\mathrm{B{\scriptsize LEU}} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

		1-gram	2-gram	3-gram
hypothesis 1	I am exhausted	3/3	1/2	0/1
hypothesis 2	Tired is I	1/3	0/2	0/1
hypothesis 3	I I I	1/3	0/2	0/1

reference 1 I am tired

reference 2 I am ready to sleep now and so exhausted

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\mathrm{B{\scriptstyle LEU}} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

- r = length of reference
  c = length of prediction

Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

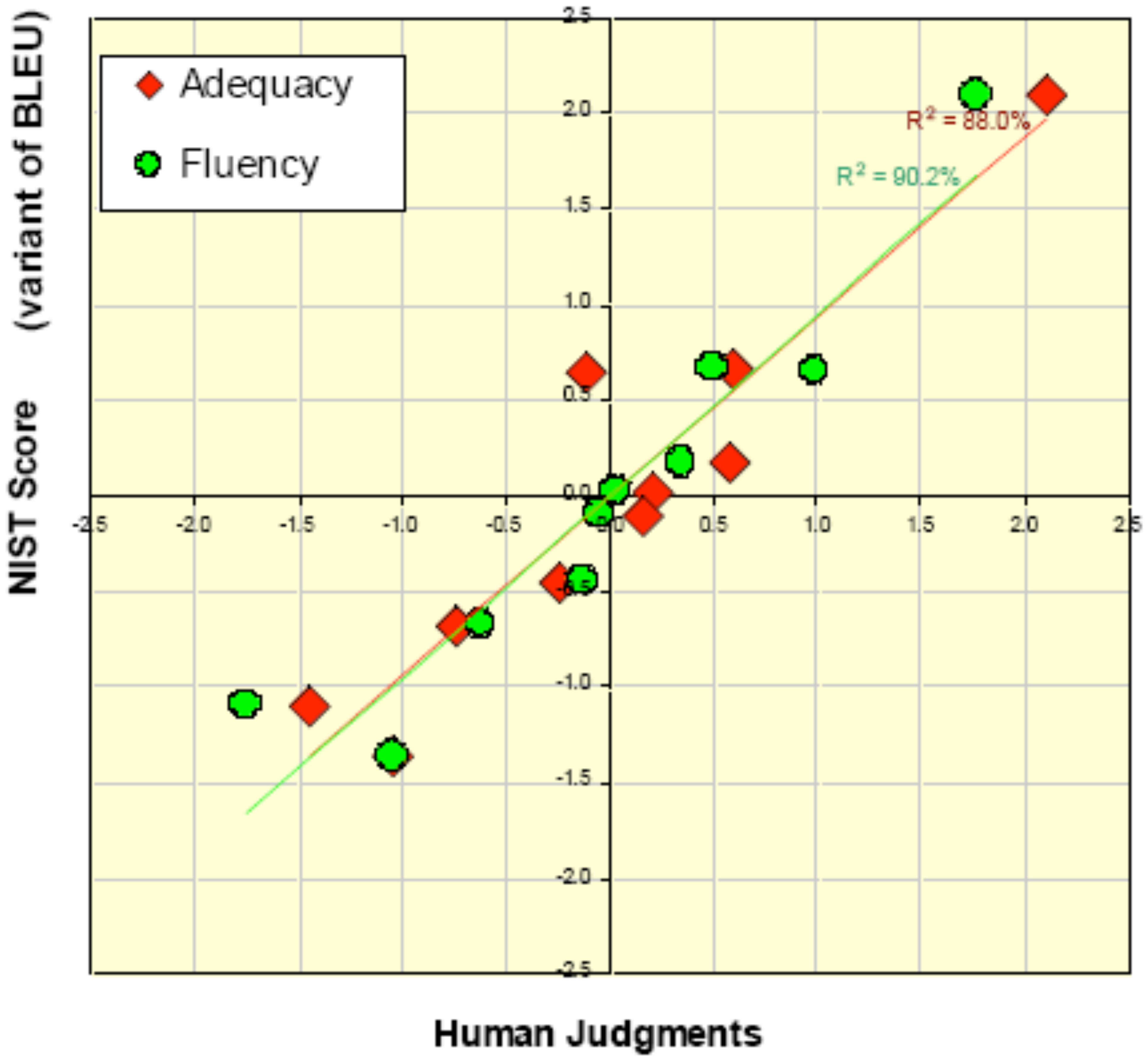
$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

- r = length of reference
  c = length of prediction

- Does this capture fluency and adequacy?

BLEU Score

- Better methods with human-in-the-loop
- HTER: human-assisted translation error rate
- If you're building real MT systems, you do user studies. In academia, you mostly use BLEU



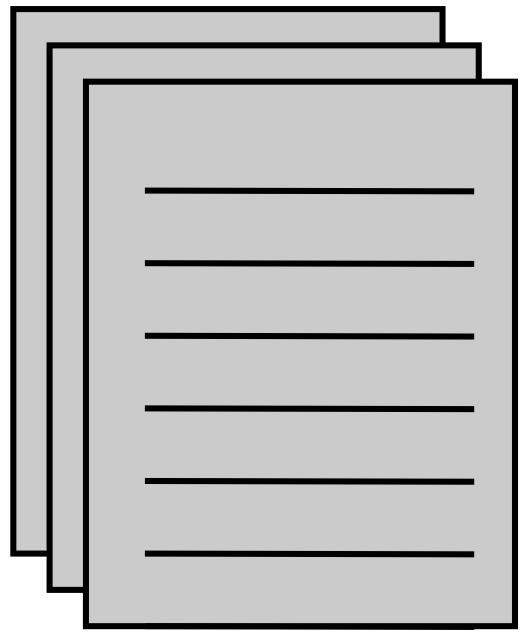
slide from G. Doddington (NIST)

Language Modeling

Phrase-Based MT

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Language model P(e)

Unlabeled English data

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

“Translate faithfully but make fluent English”

N-gram Language Models

I visited San _____ put a distribution over the next word

N-gram Language Models

I visited San ______   put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

Maximum likelihood estimate of this probability from a corpus

N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

Maximum likelihood estimate of this probability from a corpus

- Just relies on counts, even in 2008 could scale up to 1.3M word types, 4B n-grams (all 5-grams occurring >40 times on the Web)

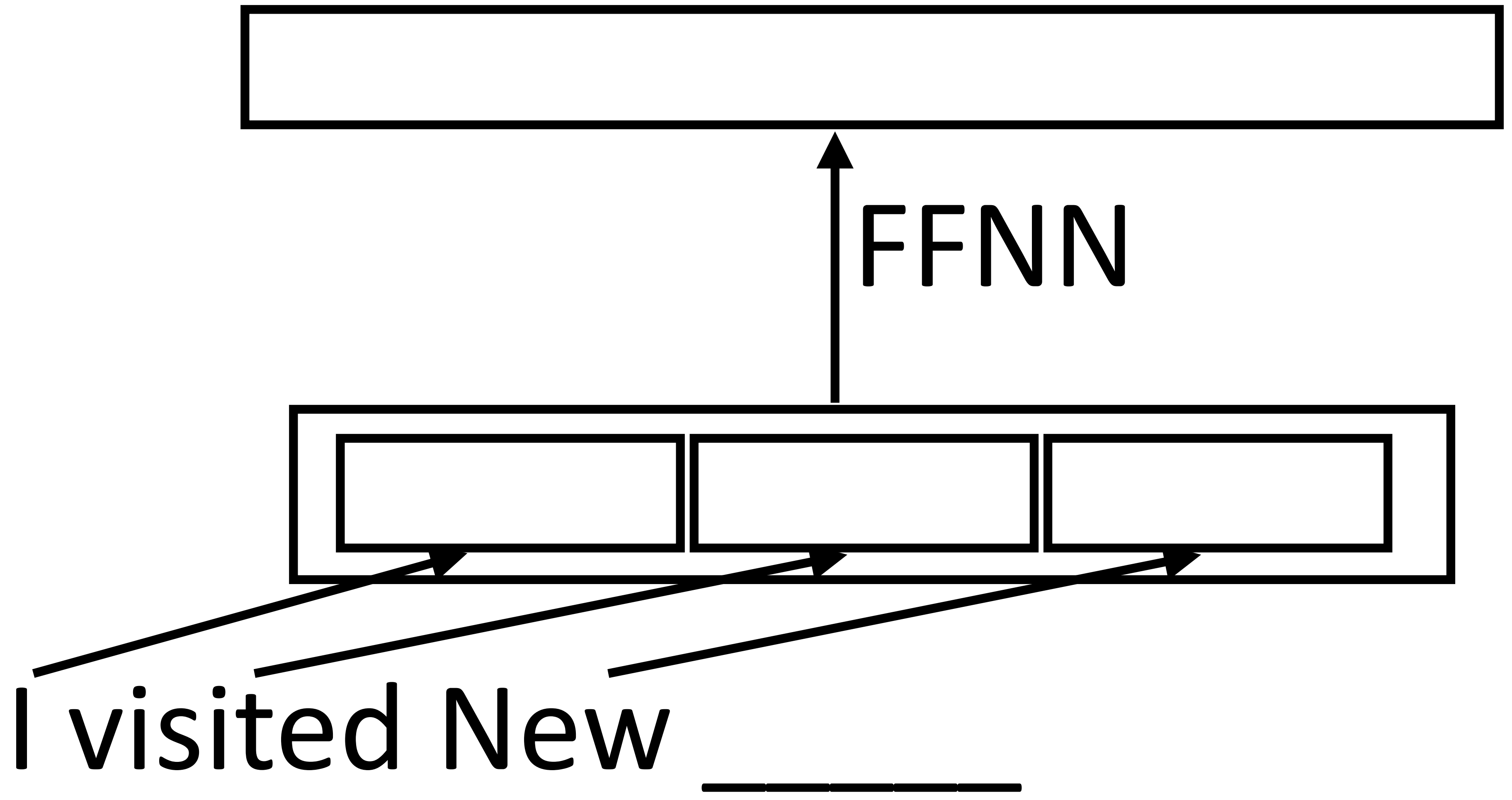
Neural Language Models

- Early work: feedforward neural networks looking at context

Mnih and Hinton (2003)

Neural Language Models

- Early work: feedforward neural networks looking at context



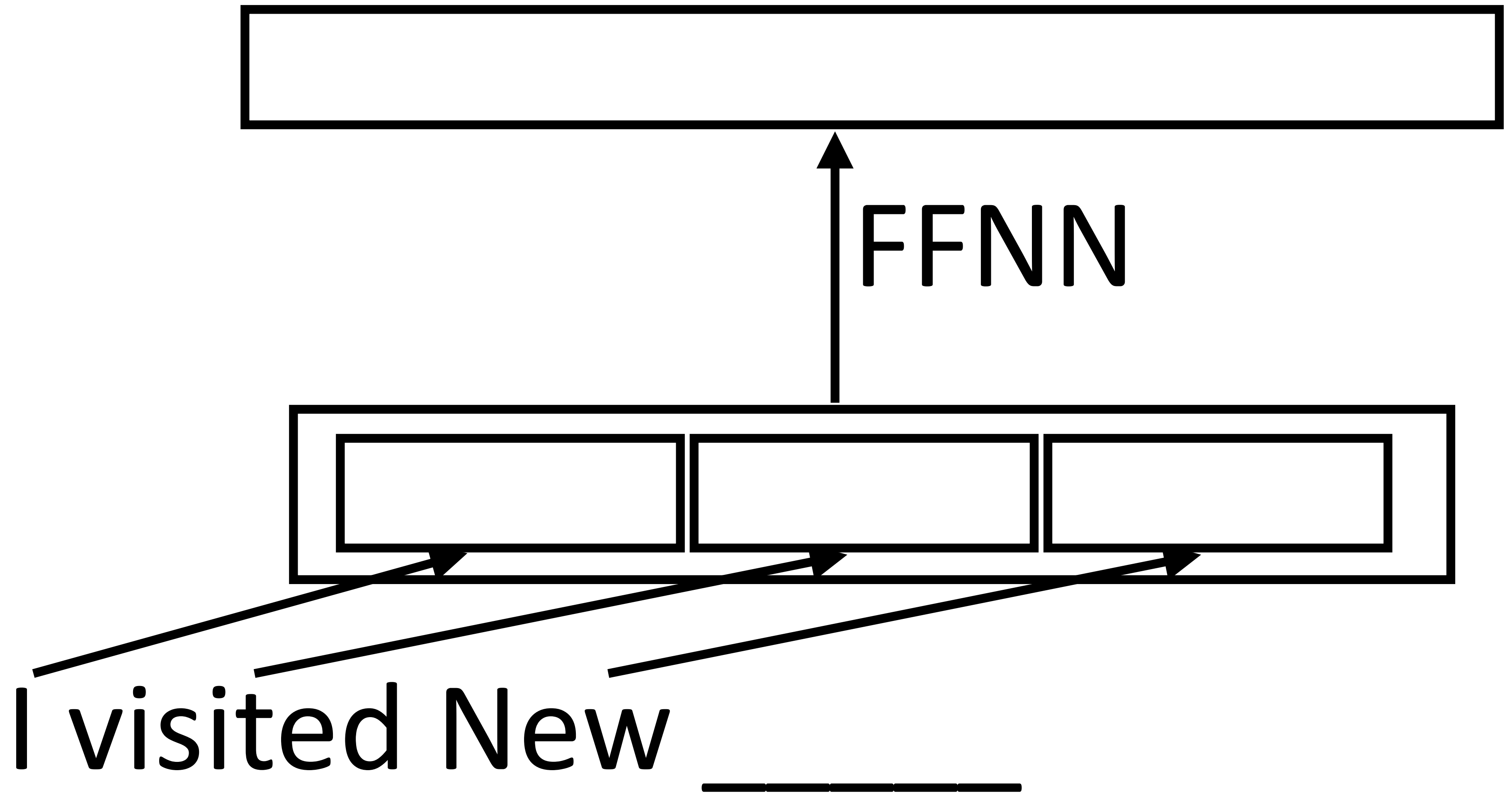
$$P(w_i | w_{i-n}, \ldots, w_{i-1})$$

Mnih and Hinton (2003)

Neural Language Models

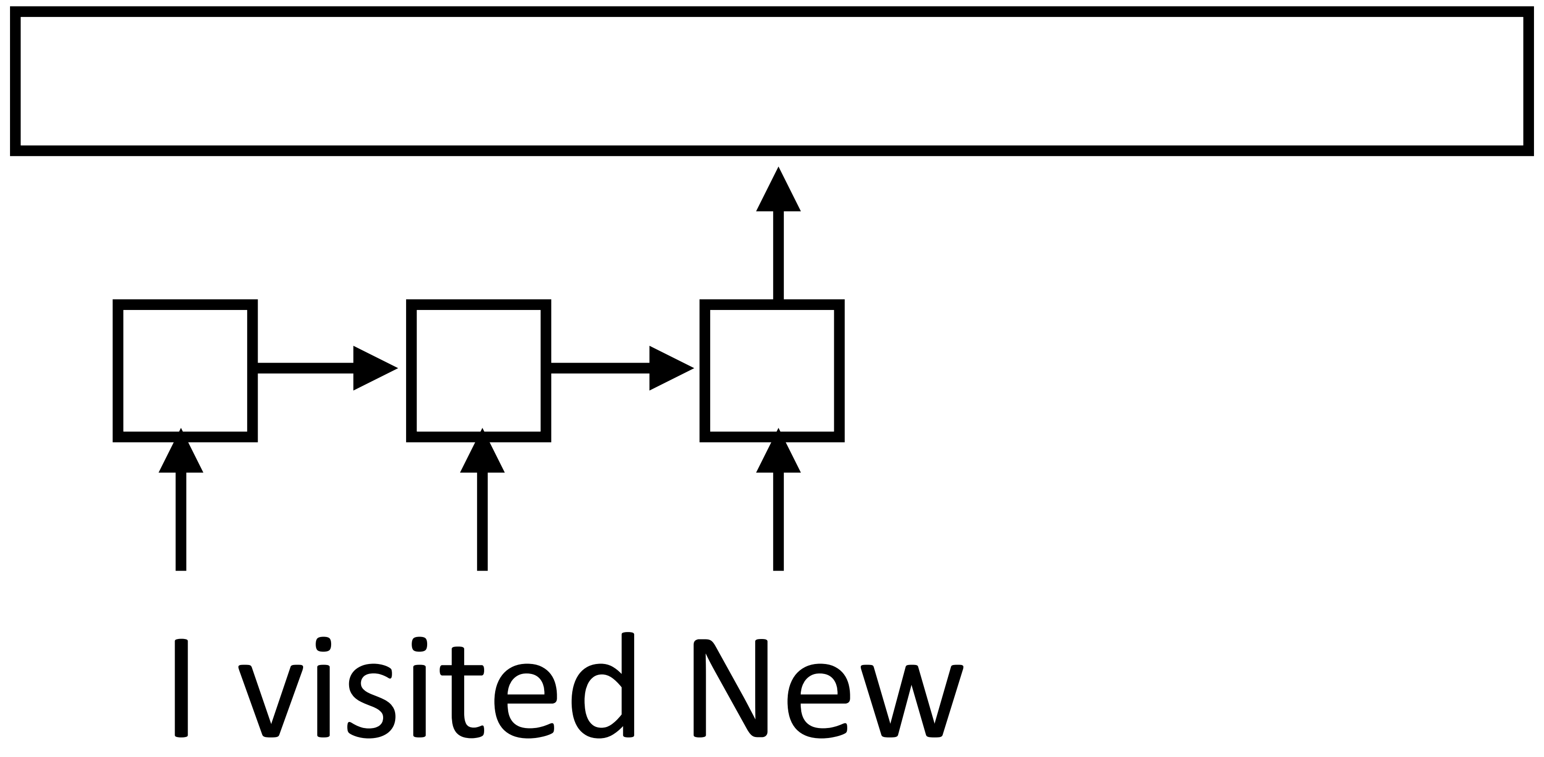
- Early work: feedforward neural networks looking at context

$$P(w_i|w_{i-n},\ldots,w_{i-1})$$



- Variable length context with RNNs:

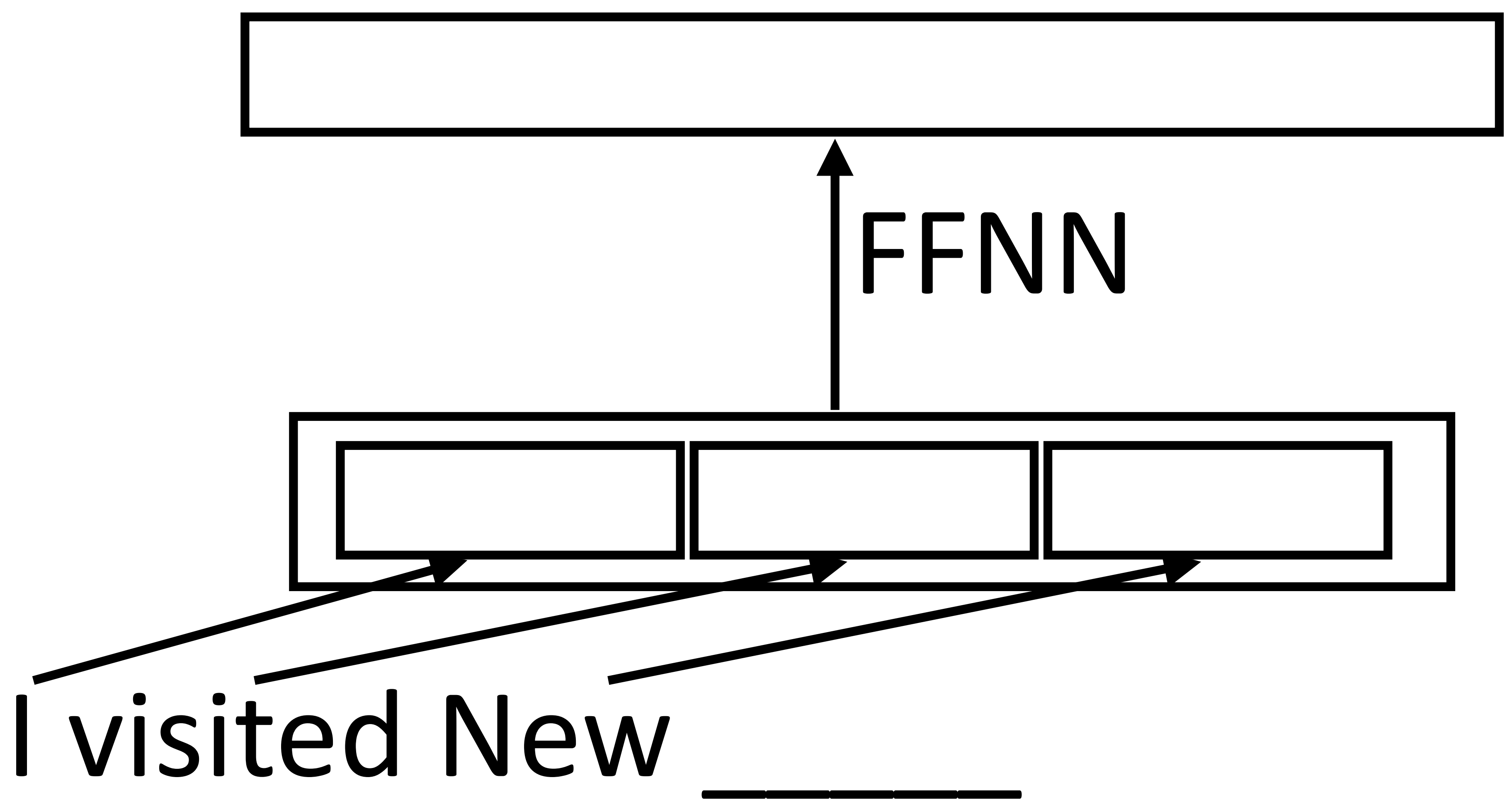
$$P(w_i|w_1,\ldots,w_{i-1})$$



Mnih and Hinton (2003)

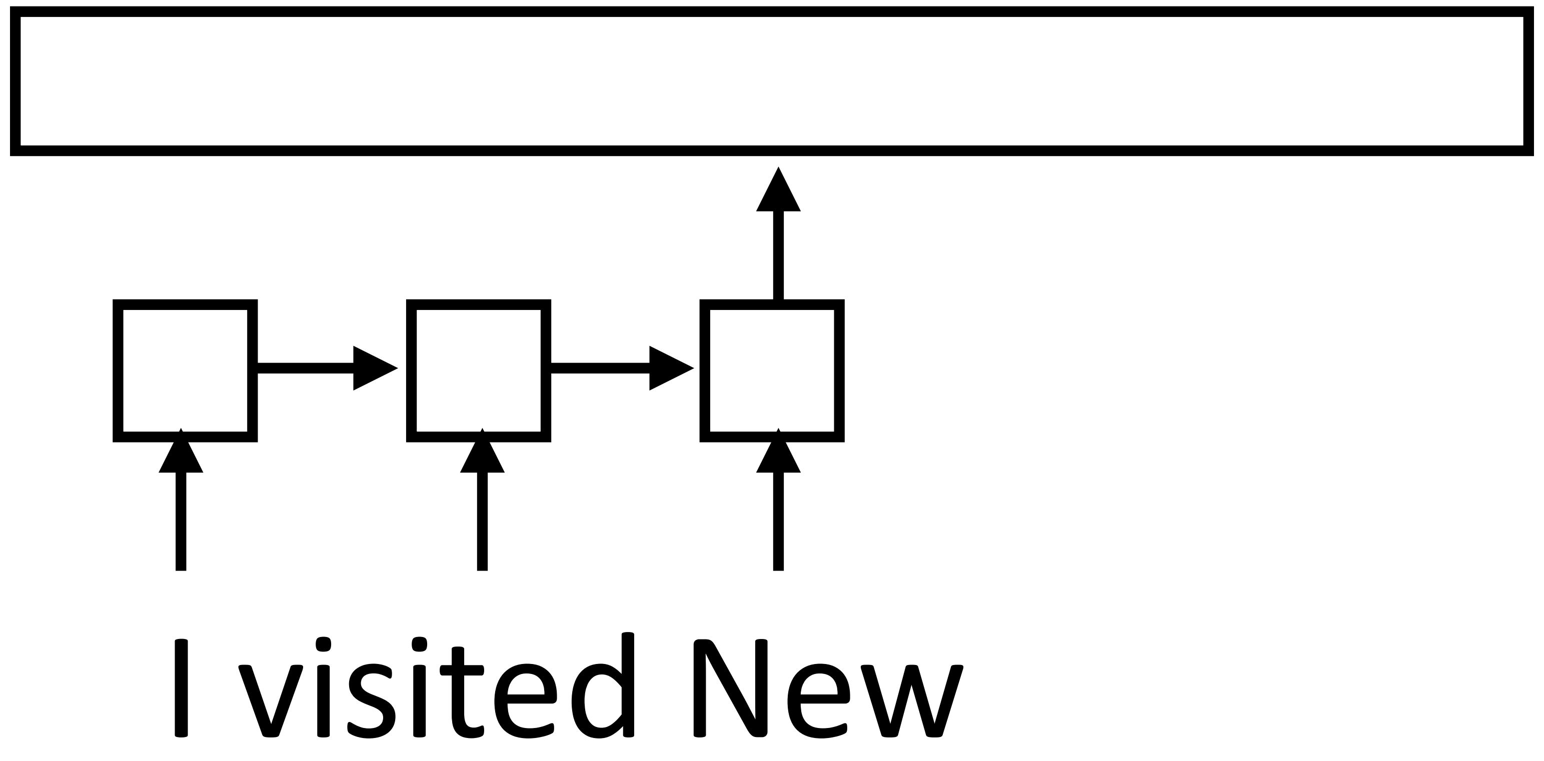
Neural Language Models

- Early work: feedforward neural networks looking at context



$$P(w_i | w_{i-n}, \ldots, w_{i-1})$$

$$P(w_i | w_1, \ldots, w_{i-1})$$

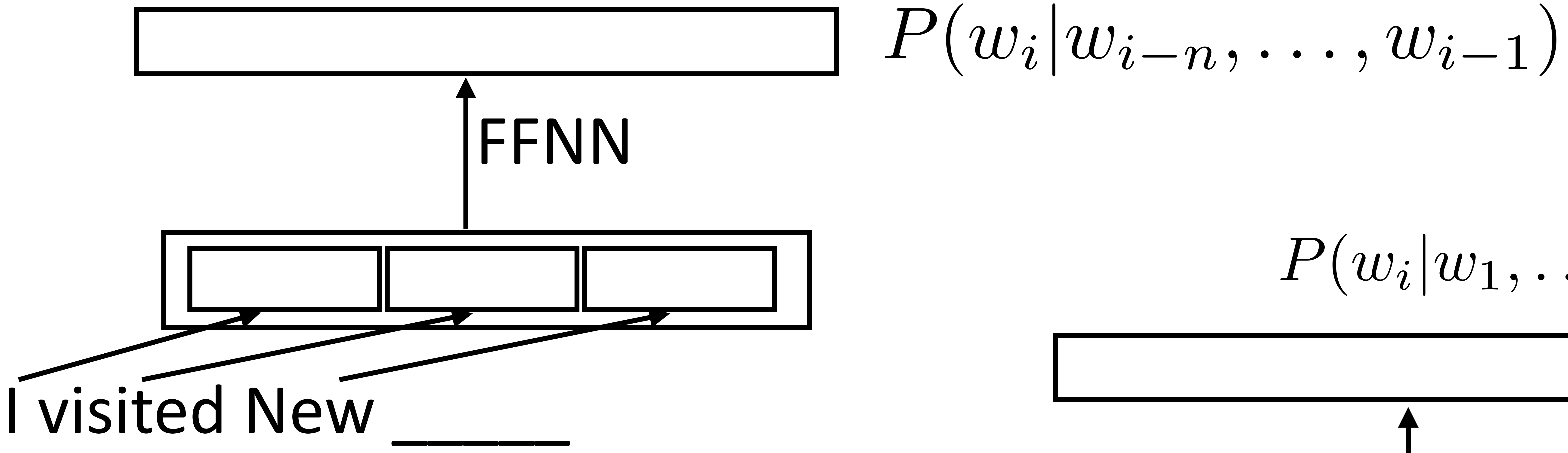


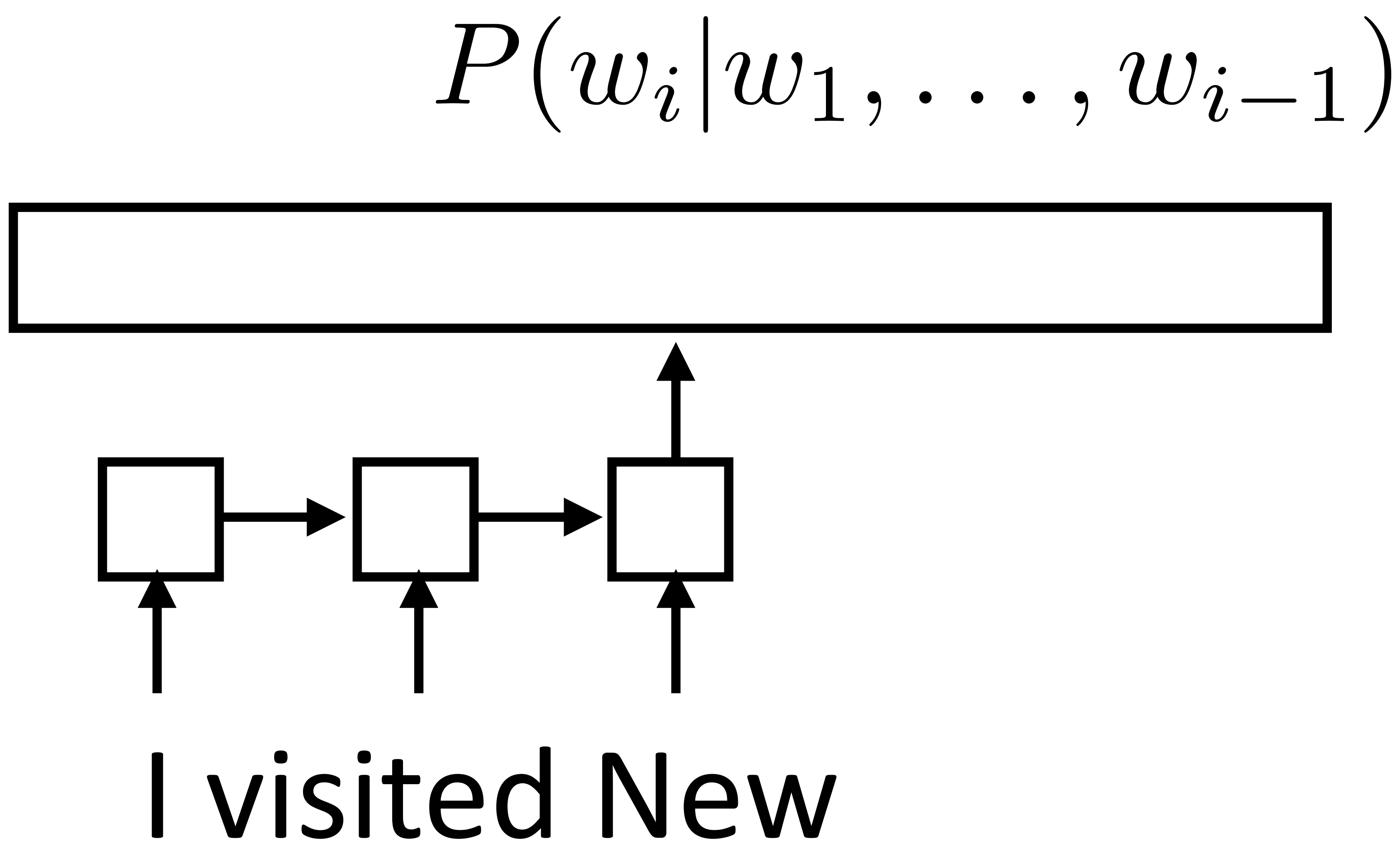
- Variable length context with RNNs:
  - Works like a decoder with no encoder

Mnih and Hinton (2003)

Neural Language Models

- Early work: feedforward neural networks looking at context





- Variable length context with RNNs:
  - Works like a decoder with no encoder
- Slow to train over lots of data!

Mnih and Hinton (2003)

Evaluation

Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i|x_1, \ldots, x_{i-1})$

Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$

- Perplexity: $2^{-\frac{1}{n}\sum_{i=1}^{n} \log_2 p(x_i | x_1, \ldots, x_{i-1})}$

Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$
- Perplexity: $2^{-\frac{1}{n}\sum_{i=1}^{n} \log_2 p(x_i|x_1,\ldots,x_{i-1})}$
  - NLL (base 2) averaged over the sentence, exponentiated

Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$

- Perplexity: $2^{-\frac{1}{n} \sum_{i=1}^{n} \log_2 p(x_i | x_1, \ldots, x_{i-1})}$

  - NLL (base 2) averaged over the sentence, exponentiated

  - NLL = -2 -> on average, correct thing has prob 1/4 -> PPL = 4. PPL is sort of like branching factor

Results

Merity et al. (2017), Melis et al. (2017)

Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark

Merity et al. (2017), Melis et al. (2017)

Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7

Merity et al. (2017), Melis et al. (2017)

Results

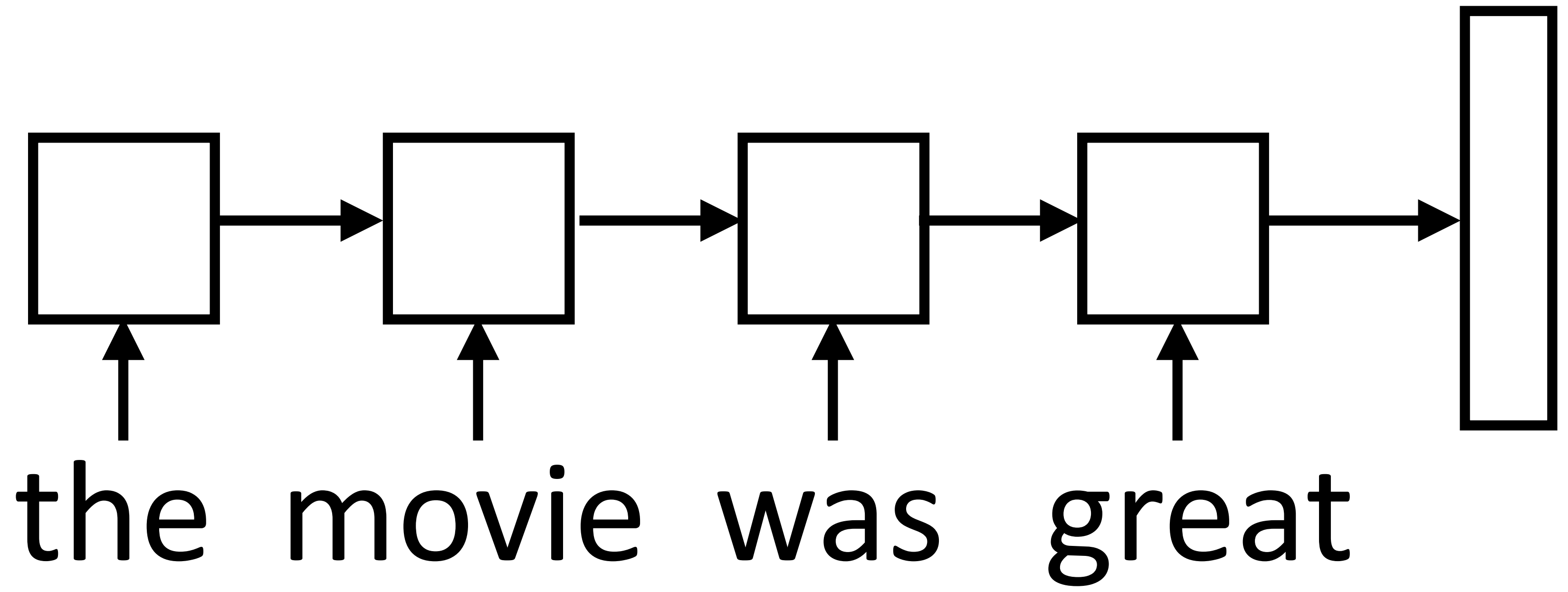
- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7
- LSTM: PPL ~ 60-80 (depending on how much you optimize it)

Merity et al. (2017), Melis et al. (2017)

Encoder-Decoder Models

Encoder-Decoder

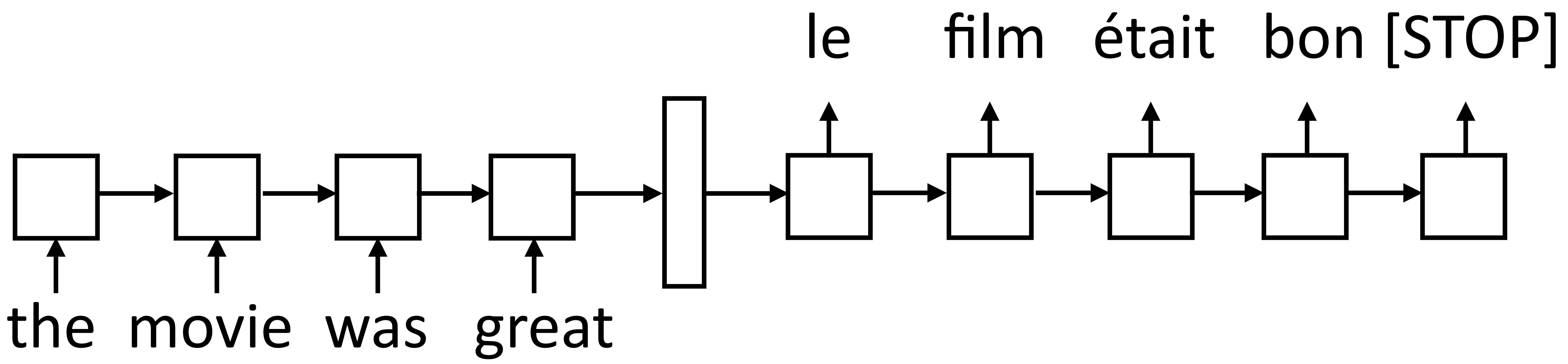
- Encode a sequence into a fixed-sized vector



Sutskever et al. (2014)

Encoder-Decoder

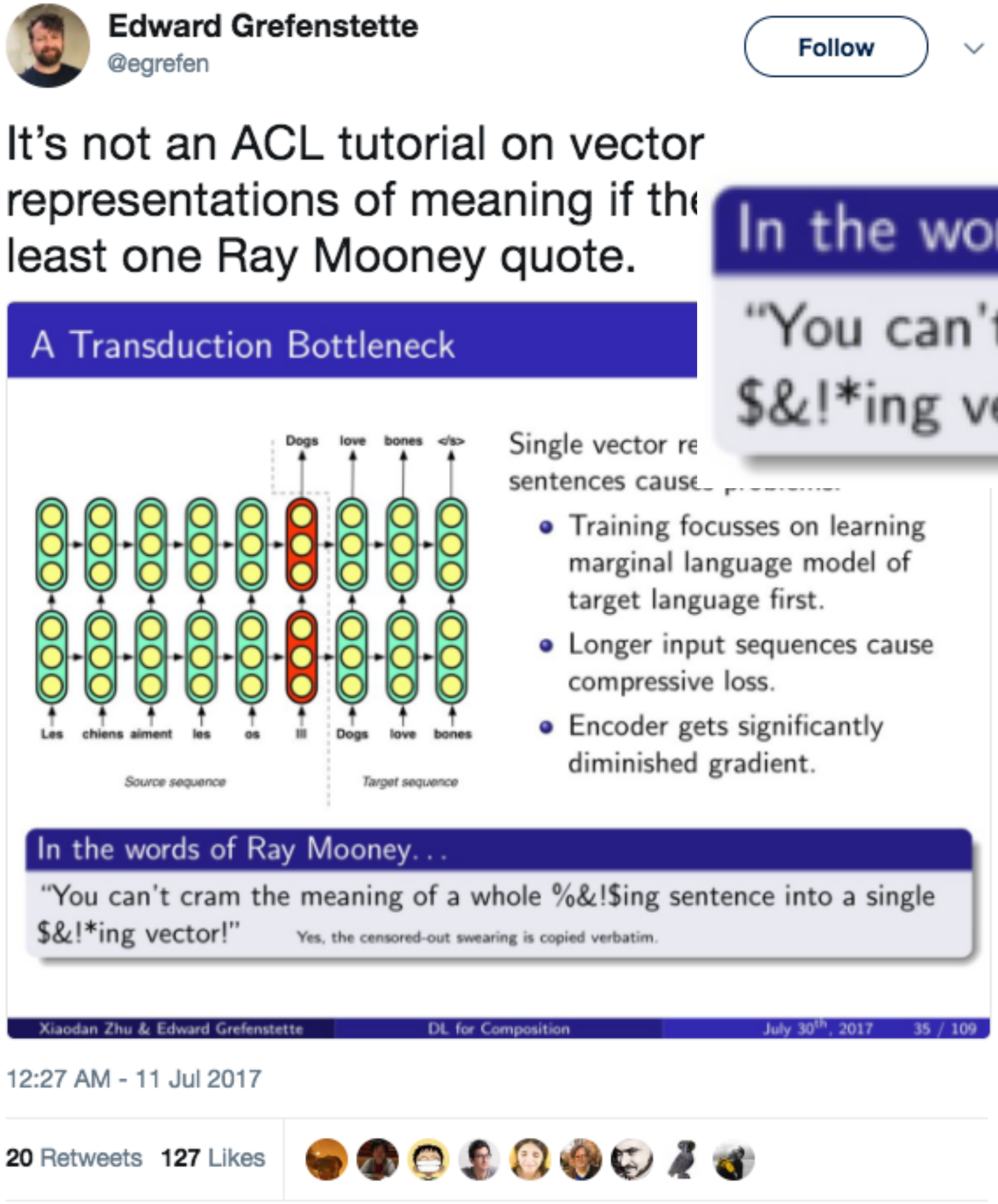
- Encode a sequence into a fixed-sized vector

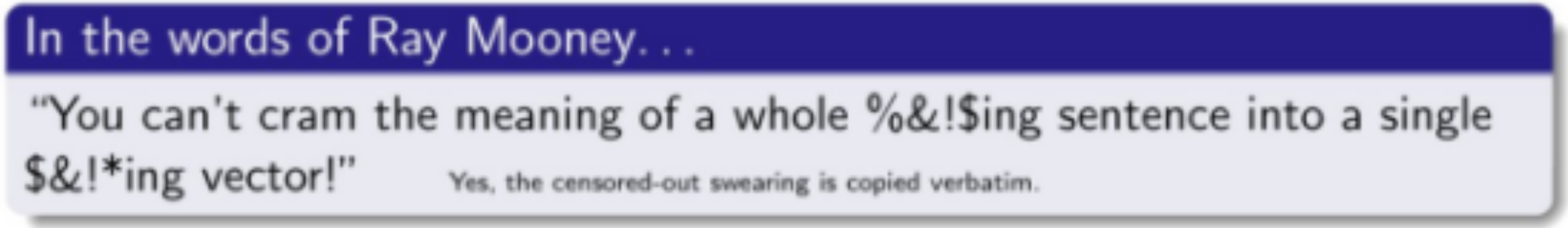


- Now use that vector to produce a series of tokens as output from a separate LSTM *decoder*

Sutskever et al. (2014)

Encoder-Decoder

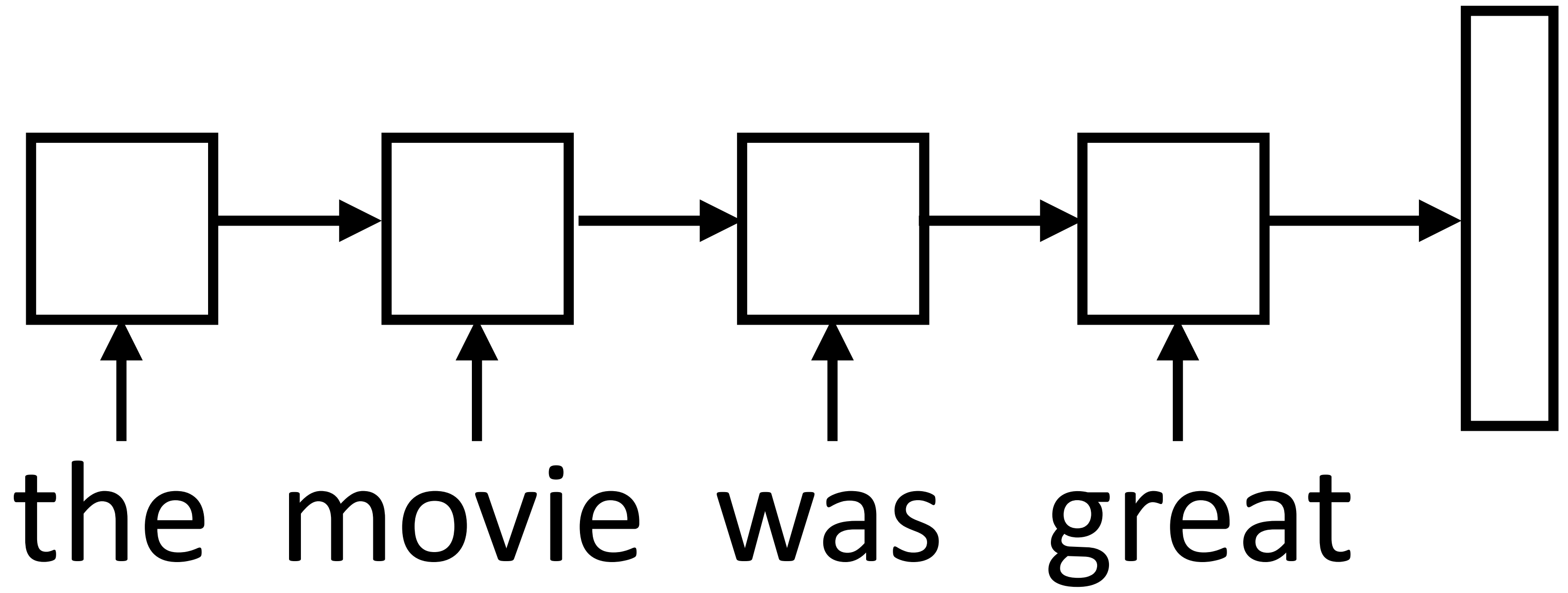




- Is this true? Sort of…we'll come back to this later

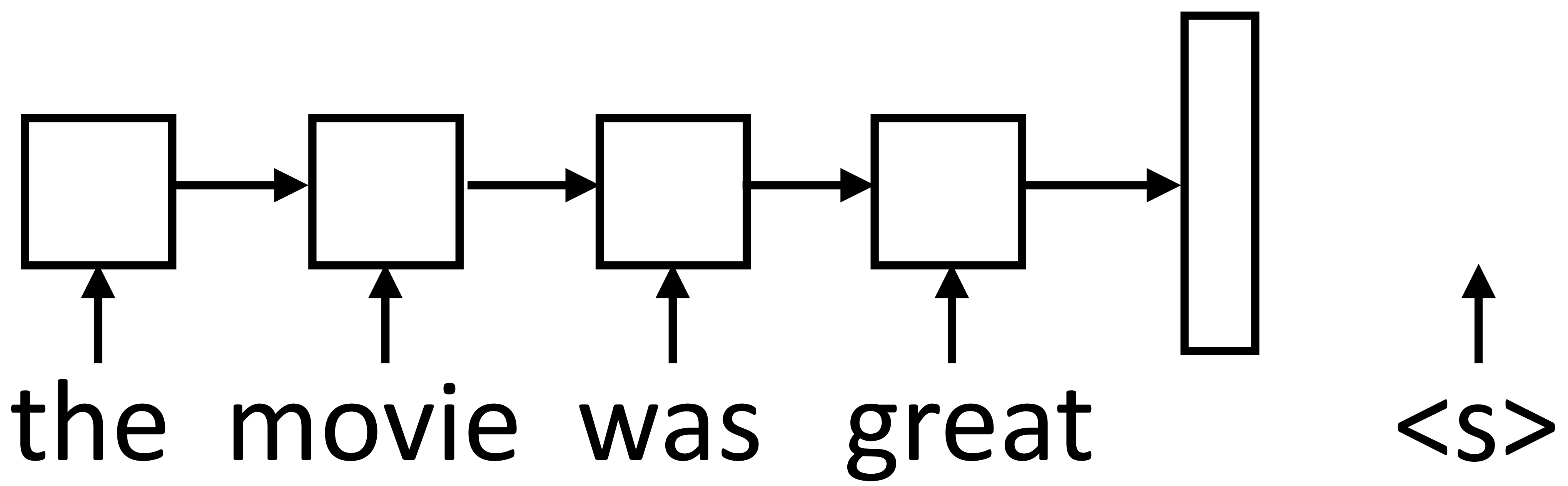
Model

- Generate next word conditioned on previous word as well as hidden state



Model

- Generate next word conditioned on previous word as well as hidden state

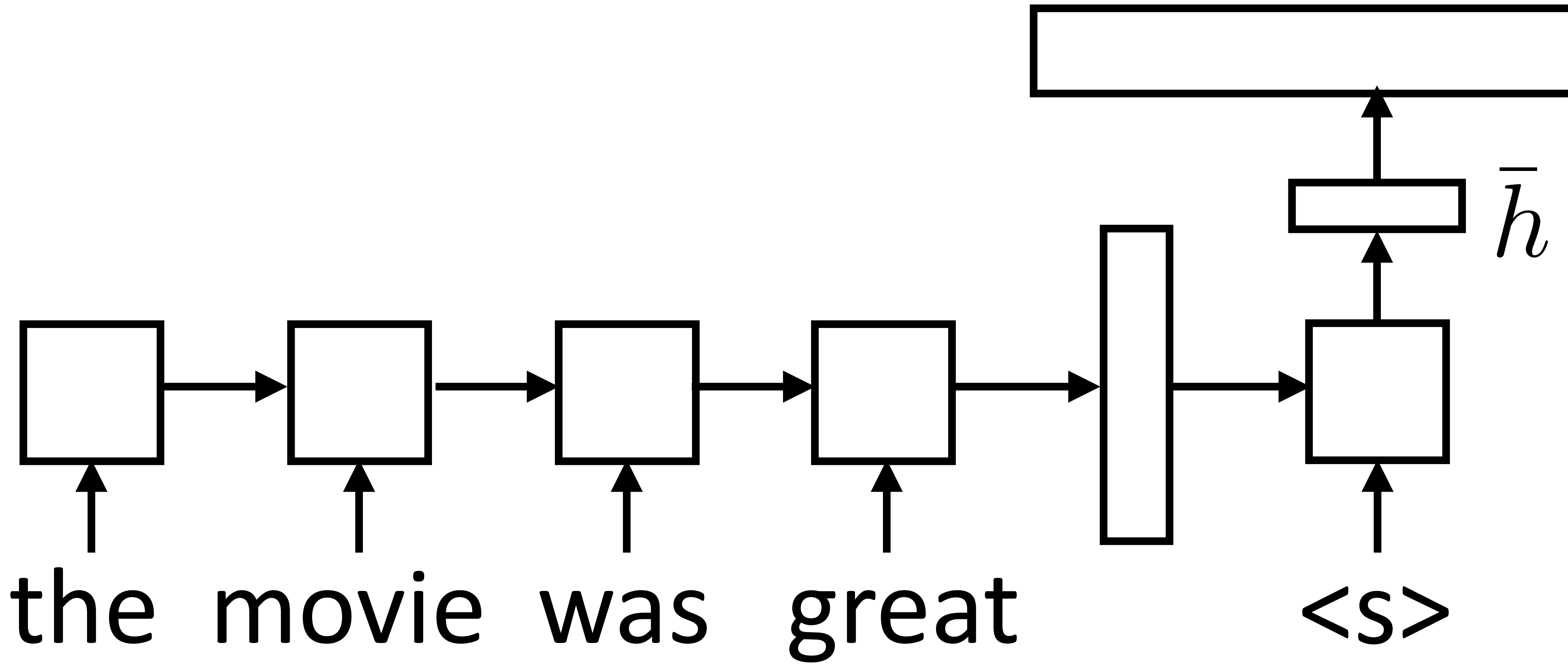


Model

- Generate next word conditioned on previous word as well as hidden state

- W size is |vocab| x |hidden state|, softmax over entire vocabulary

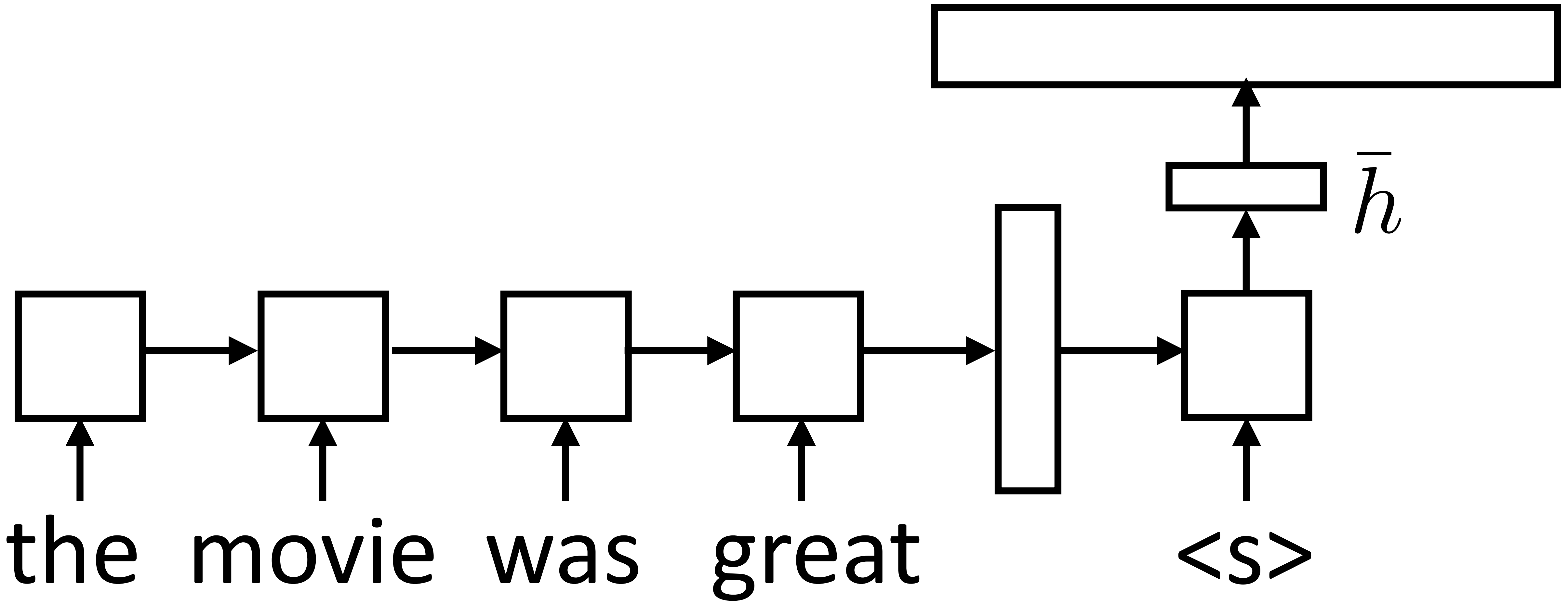
$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$



Model

- Generate next word conditioned on previous word as well as hidden state

- W size is |vocab| x |hidden state|, softmax over entire vocabulary

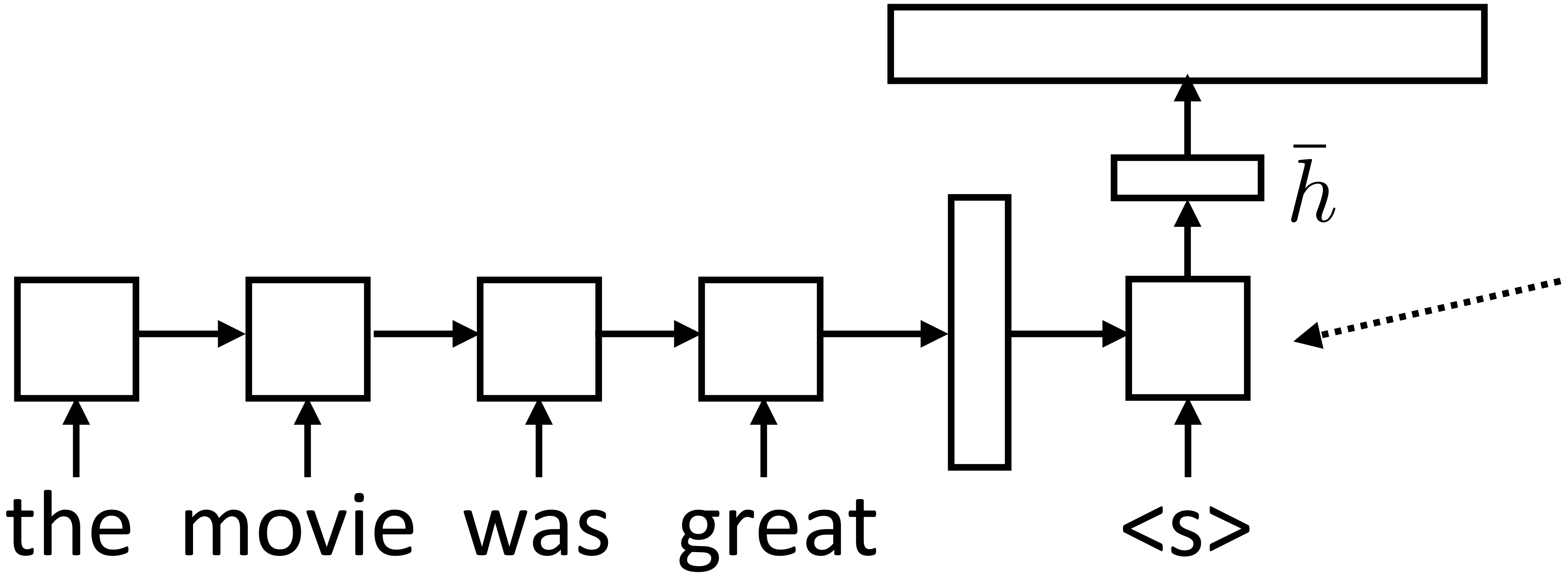


$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W\bar{h})$$

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1})$$

Model

- Generate next word conditioned on previous word as well as hidden state
- W size is |vocab| x |hidden state|, softmax over entire vocabulary



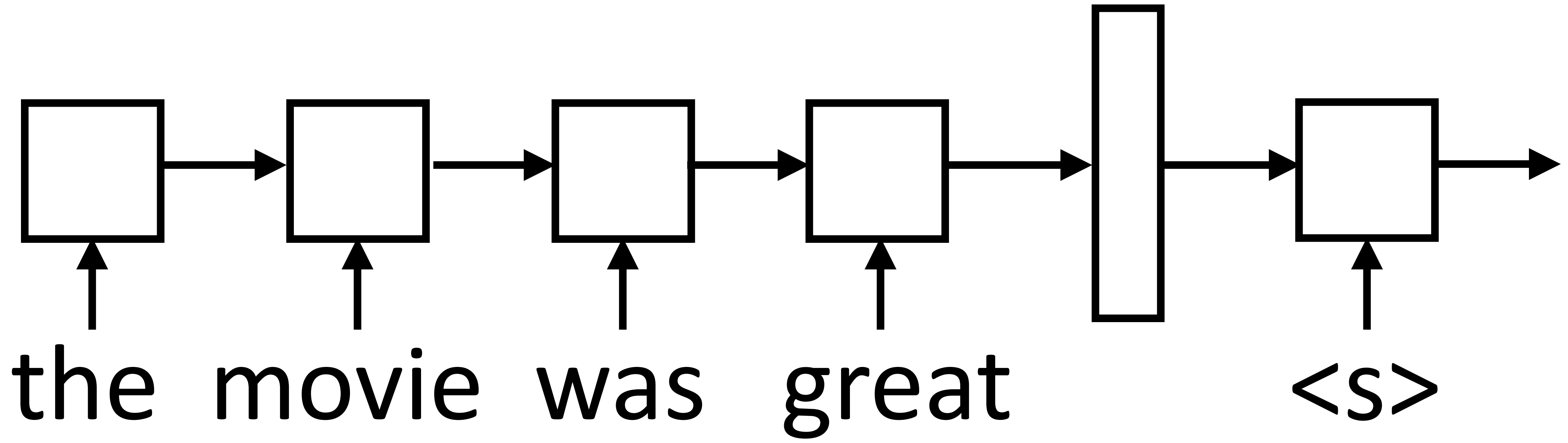
$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1})$$

Decoder has separate parameters from encoder, so this can learn to be a language model (produce a plausible next word given current one)

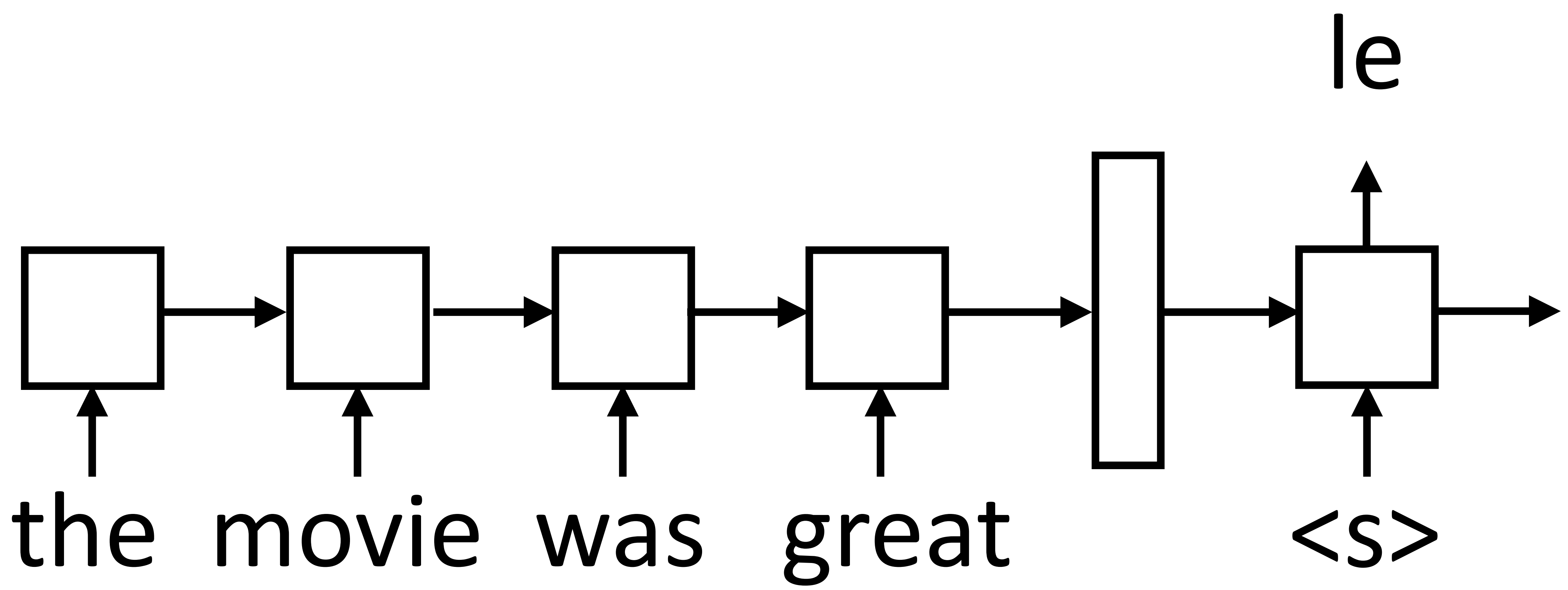
Inference

- Generate next word conditioned on previous word as well as hidden state



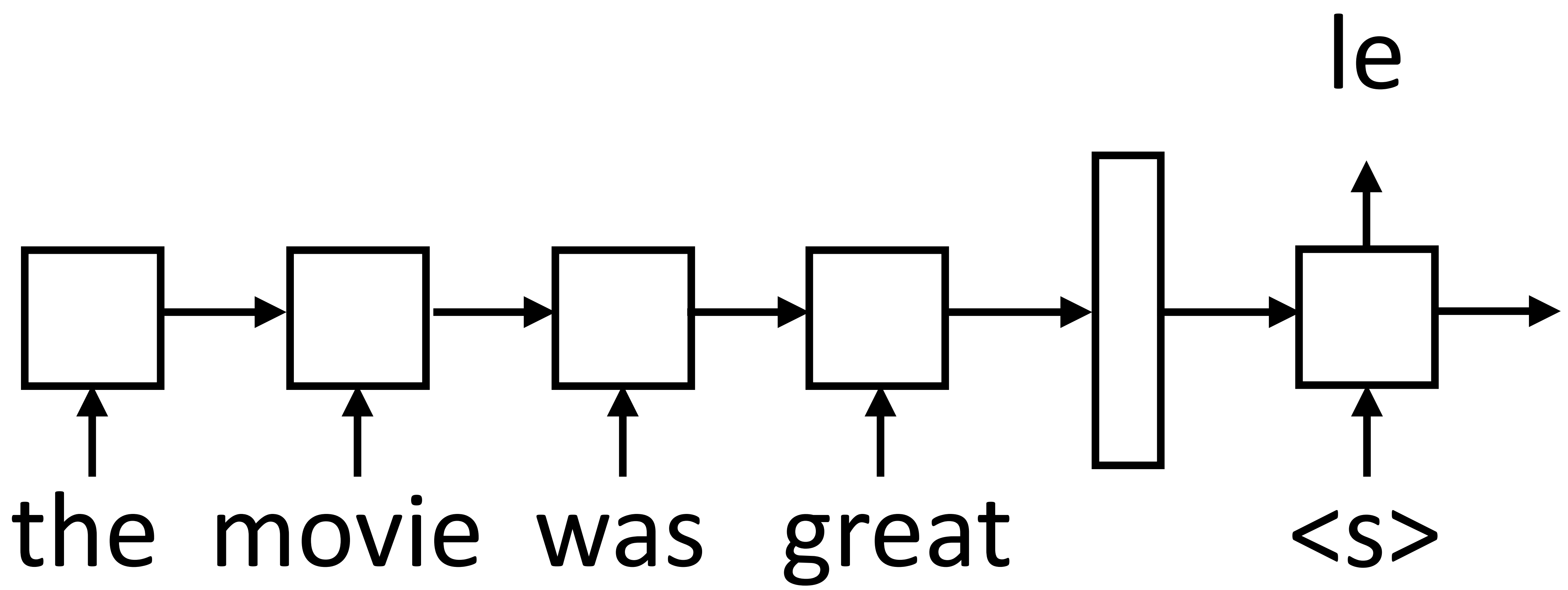
Inference

- Generate next word conditioned on previous word as well as hidden state



Inference

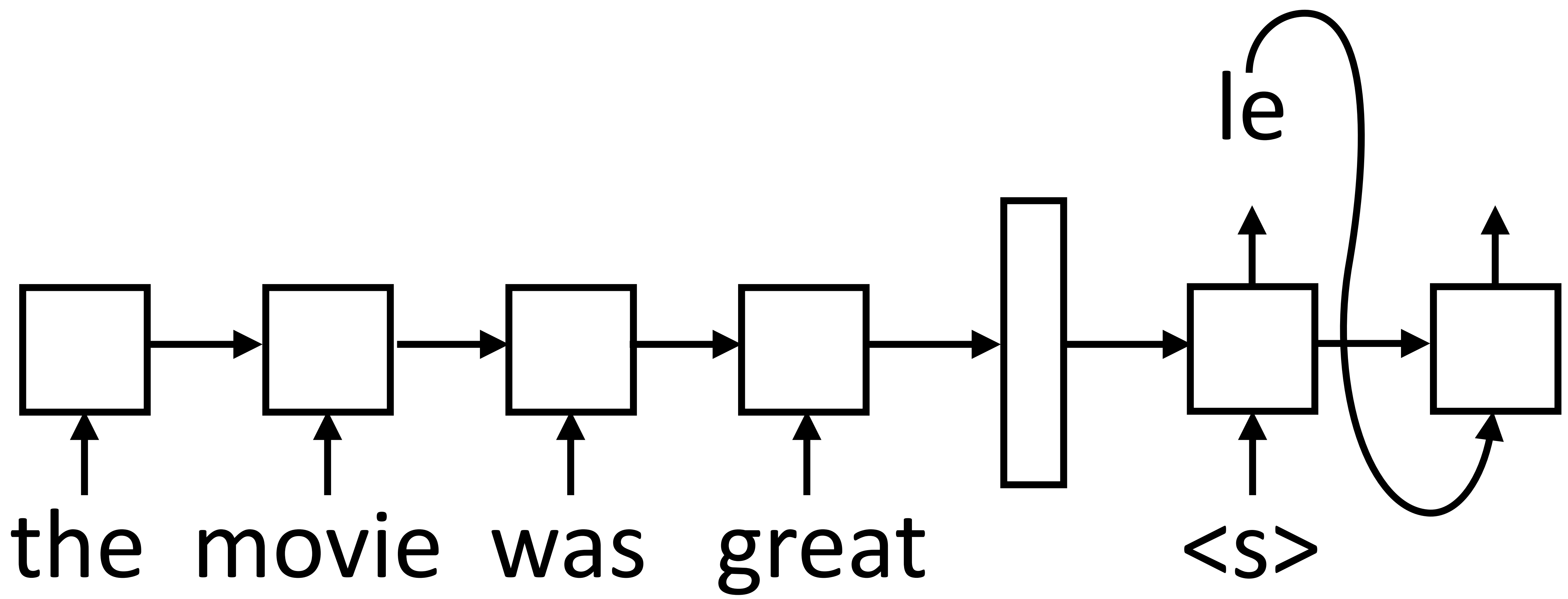
- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state

Inference

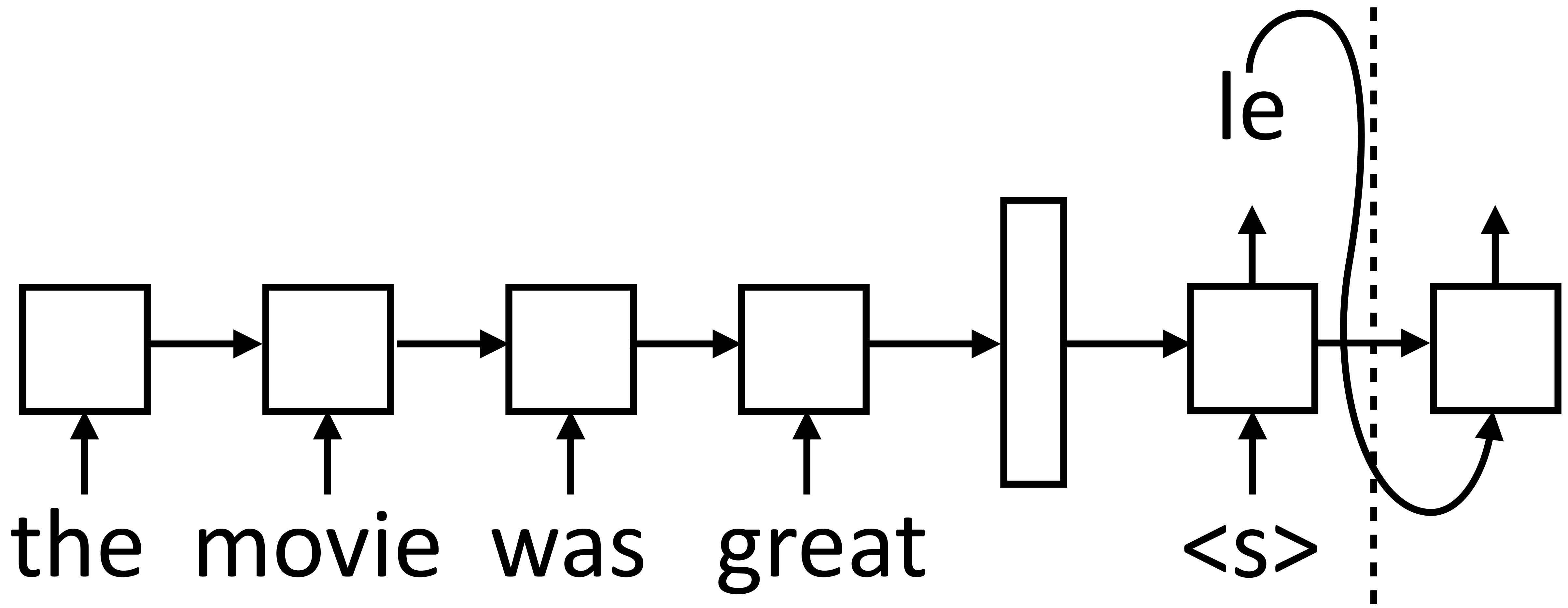
- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state

Inference

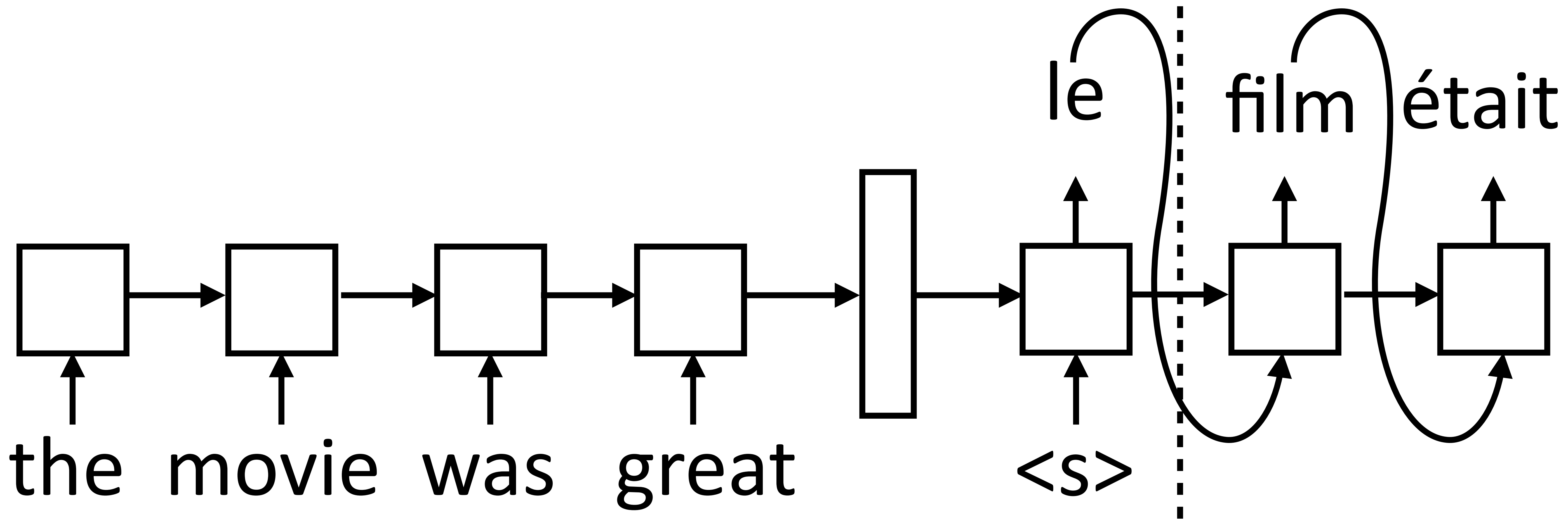
- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state

Inference

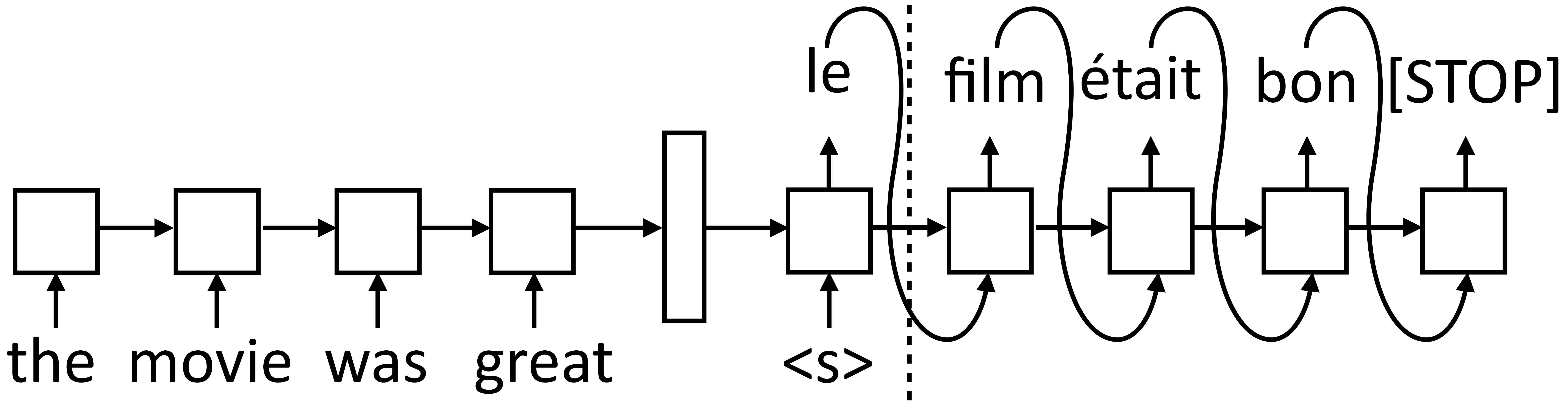
- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state

Inference

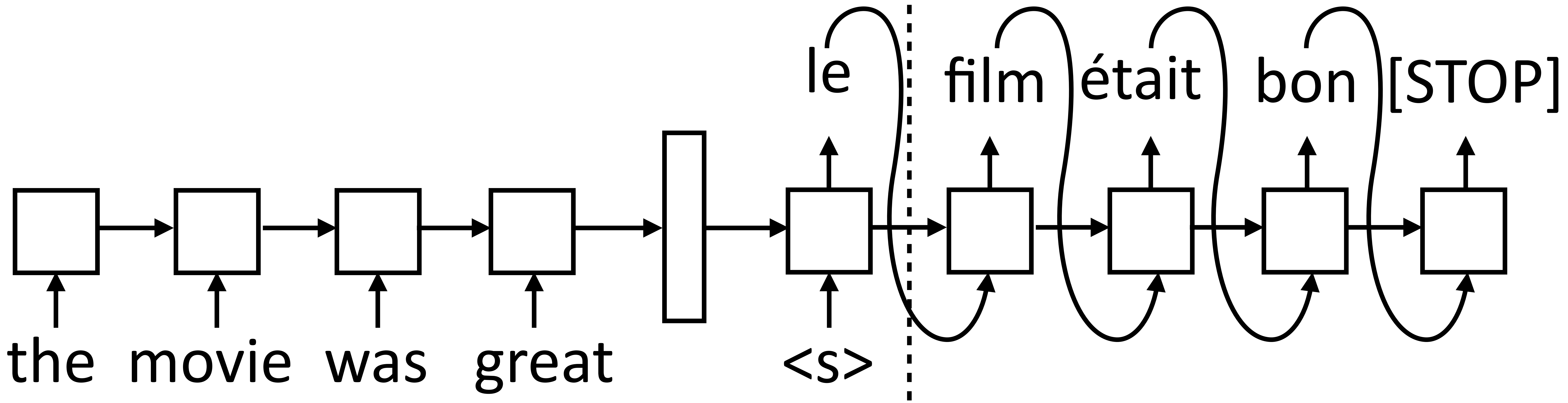
- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state

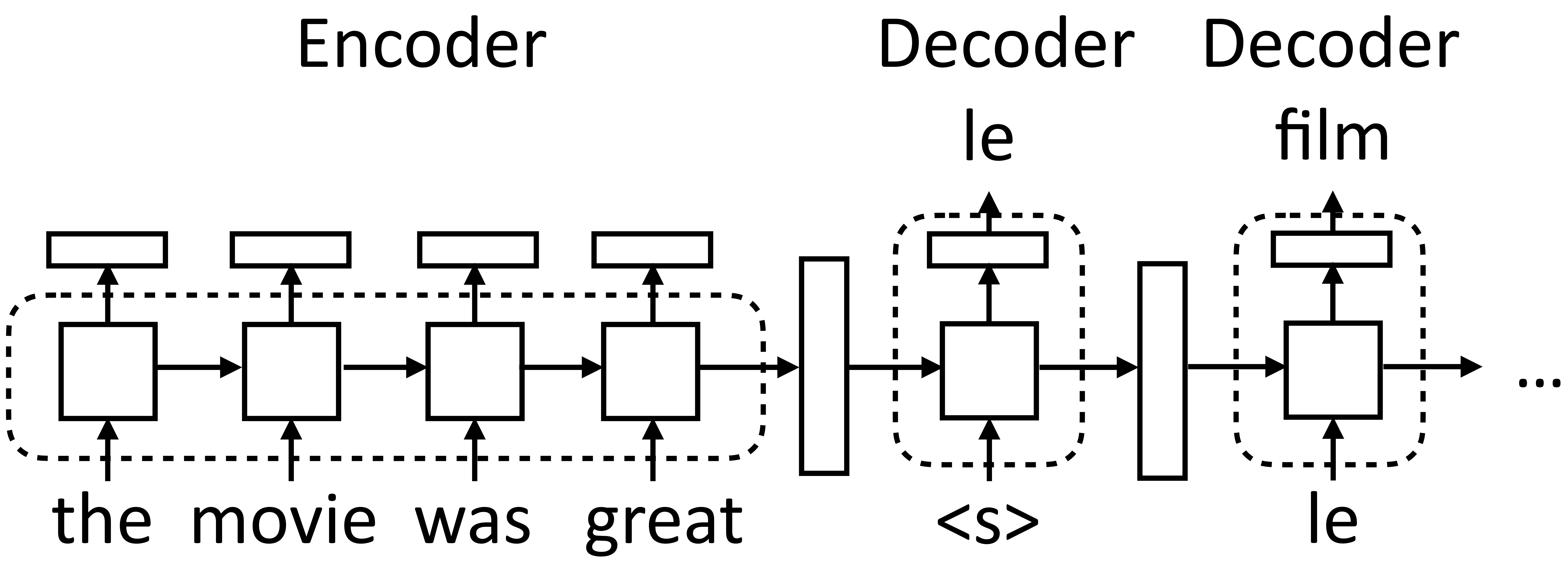
Inference

- Generate next word conditioned on previous word as well as hidden state

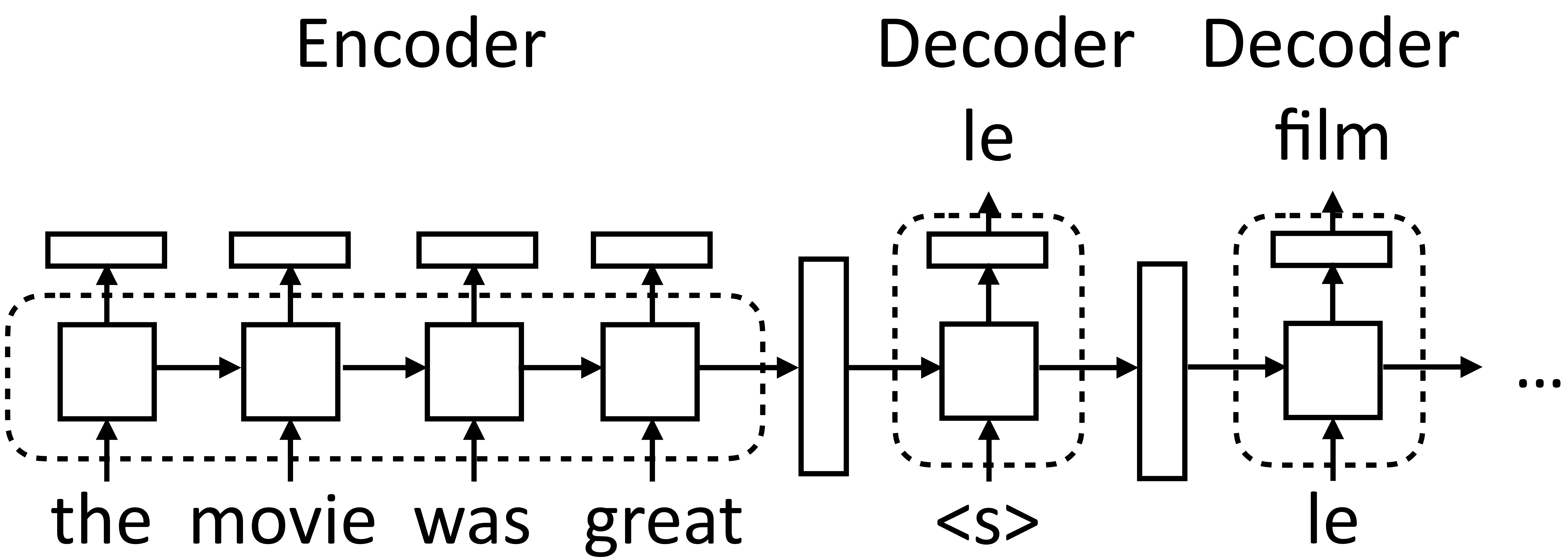


- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state
- Decoder is advanced one state at a time until [STOP] is reached

Implementing seq2seq Models

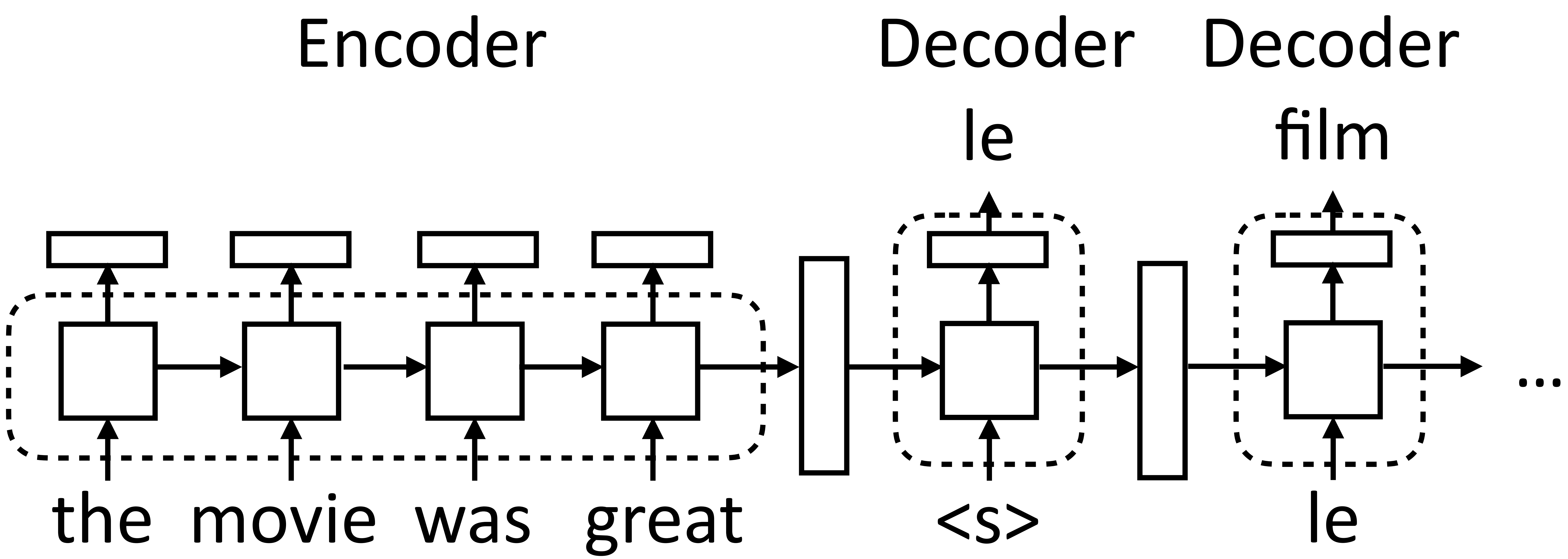


Implementing seq2seq Models



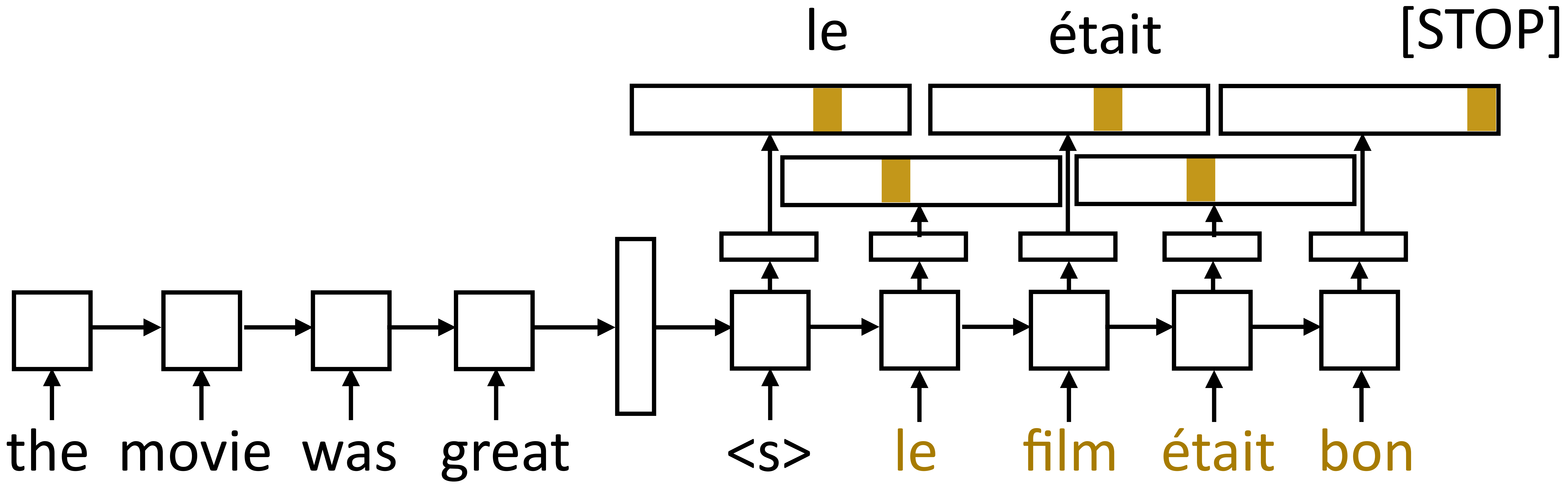
- Encoder: consumes sequence of tokens, produces a vector. Analogous to encoders for classification/tagging tasks

Implementing seq2seq Models



- Encoder: consumes sequence of tokens, produces a vector. Analogous to encoders for classification/tagging tasks
- Decoder: separate module, single cell. Takes two inputs: hidden state (vector *h* or tuple (*h*, *c*)) and previous token. Outputs token + new state

Training



- Objective: maximize $\sum_{(\mathbf{x},\mathbf{y})} \sum_{i=1}^{n} \log P(y_i^* | \mathbf{x}, y_1^*, \ldots, y_{i-1}^*)$
- One loss term for each target-sentence word, feed the correct word regardless of model's prediction

Implementation Details

Implementation Details

- Sentence lengths vary for both encoder and decoder:

Implementation Details

- Sentence lengths vary for both encoder and decoder:
  - Typically pad everything to the right length

Implementation Details

- Sentence lengths vary for both encoder and decoder:
  - Typically pad everything to the right length
- Encoder: Can be a CNN/LSTM/Transformer…

Implementation Details

- Sentence lengths vary for both encoder and decoder:
  - Typically pad everything to the right length
- Encoder: Can be a CNN/LSTM/Transformer…
- Decoder: also flexible in terms of architecture (more later). Execute one step of computation at a time, so computation graph is formulated as taking one input + hidden state

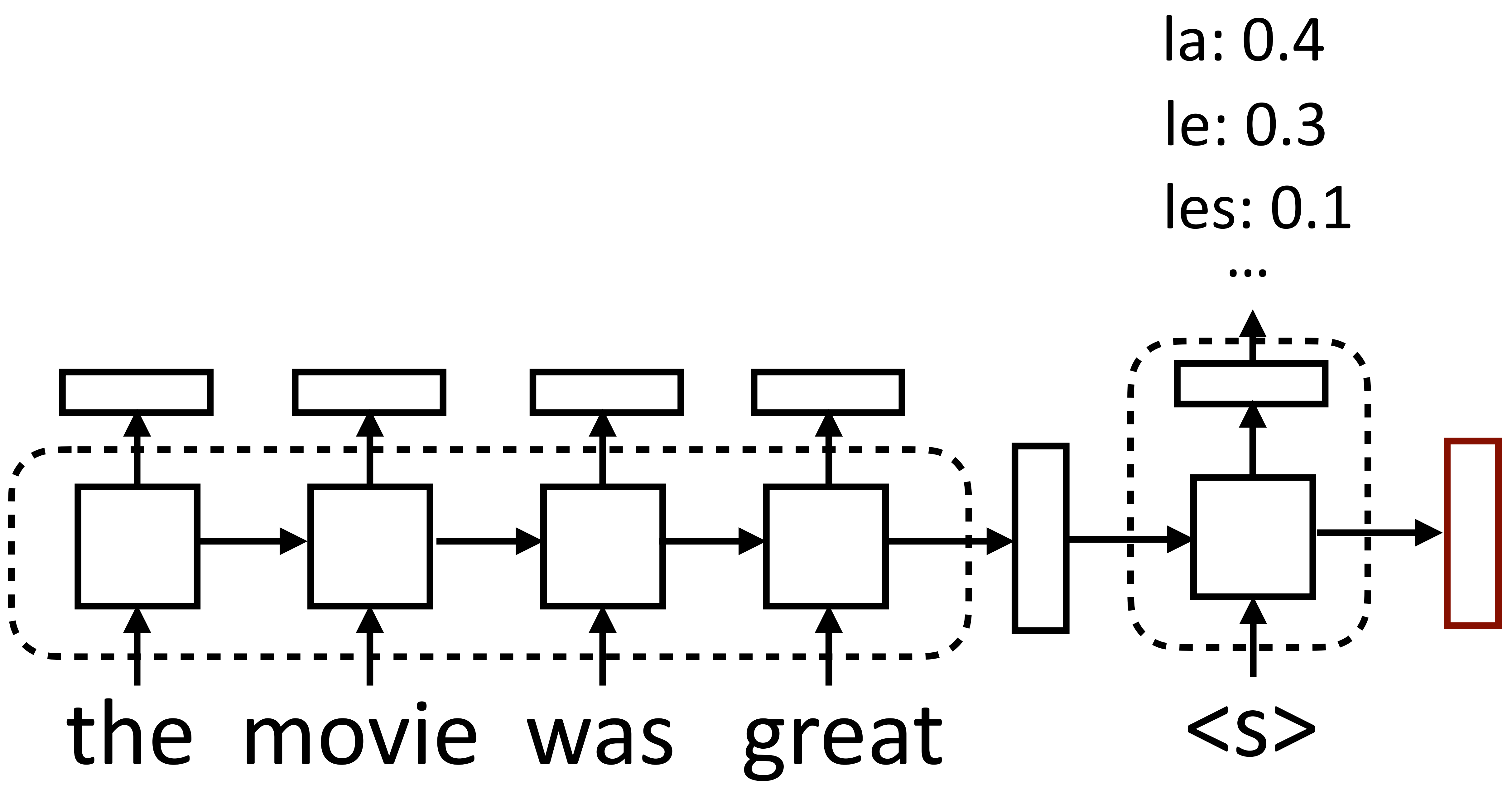
Implementation Details

- Sentence lengths vary for both encoder and decoder:
  - Typically pad everything to the right length
- Encoder: Can be a CNN/LSTM/Transformer…
- Decoder: also flexible in terms of architecture (more later). Execute one step of computation at a time, so computation graph is formulated as taking one input + hidden state
- Beam search: can help with lookahead. Finds the (approximate) highest scoring sequence:

$$\text{argmax}_{\mathbf{y}} \prod_{i=1}^{n} P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1})$$

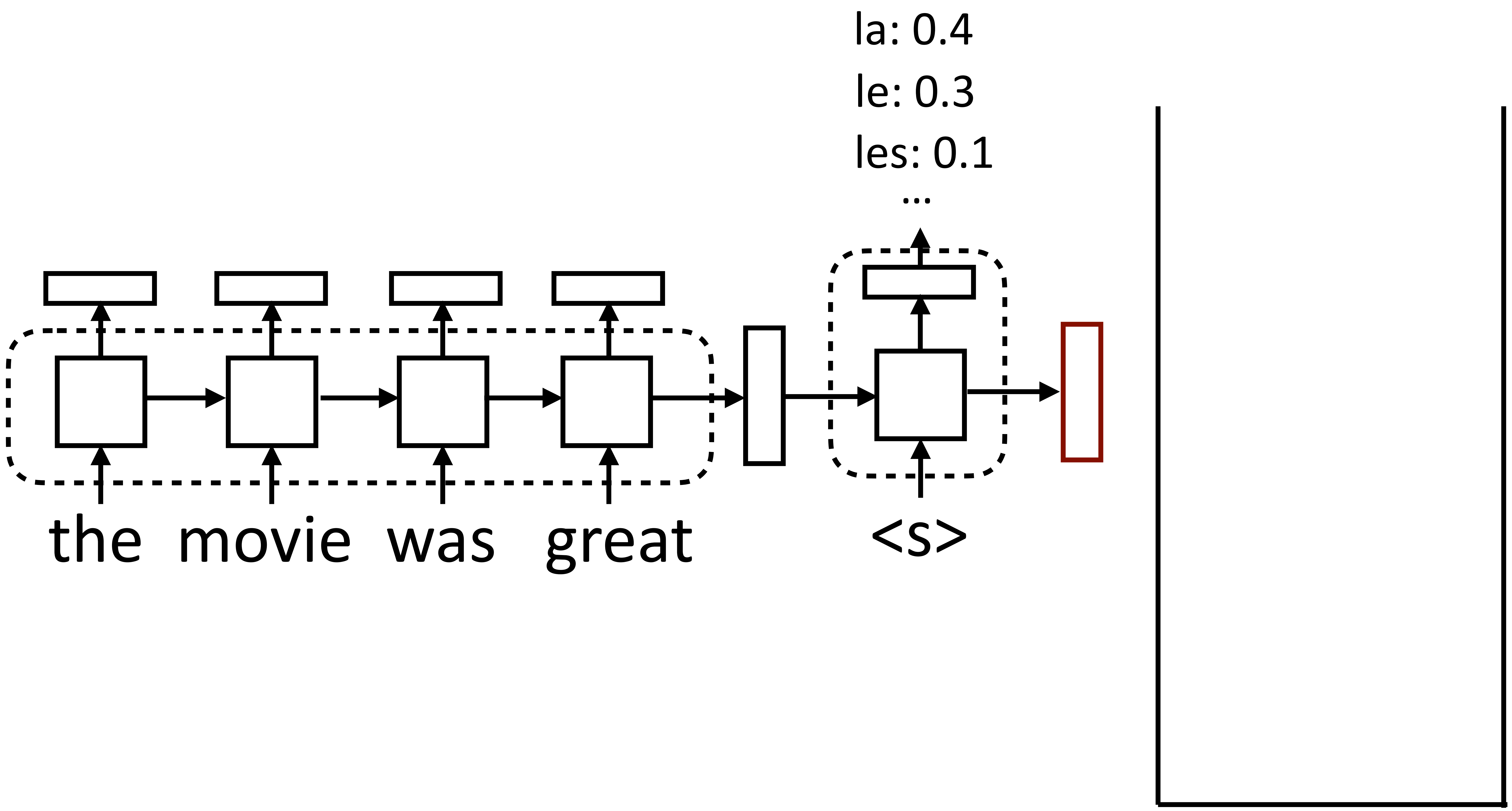
Beam Search

- Maintain decoder state, token history in beam



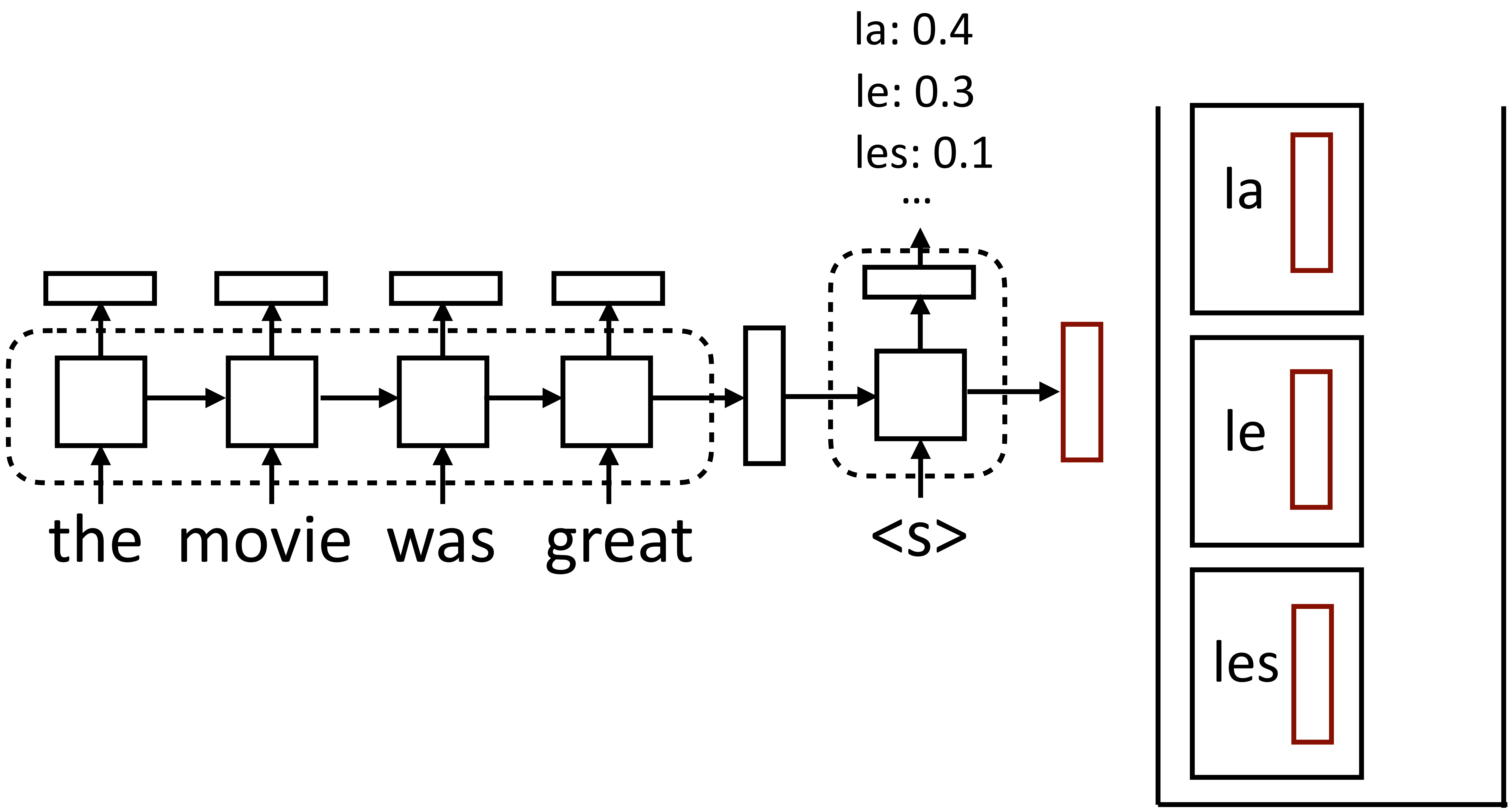
Beam Search

- Maintain decoder state, token history in beam



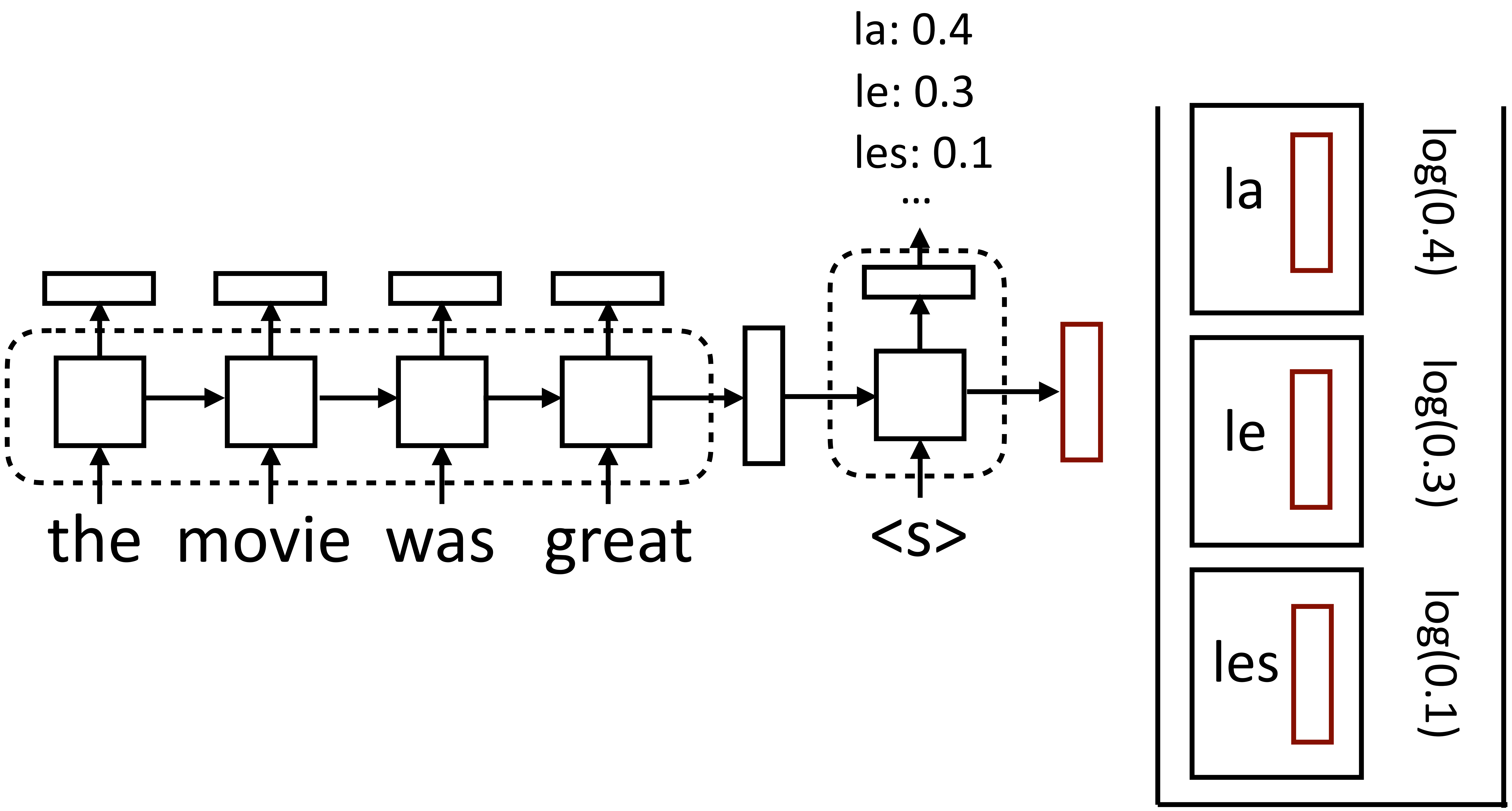
Beam Search

- Maintain decoder state, token history in beam



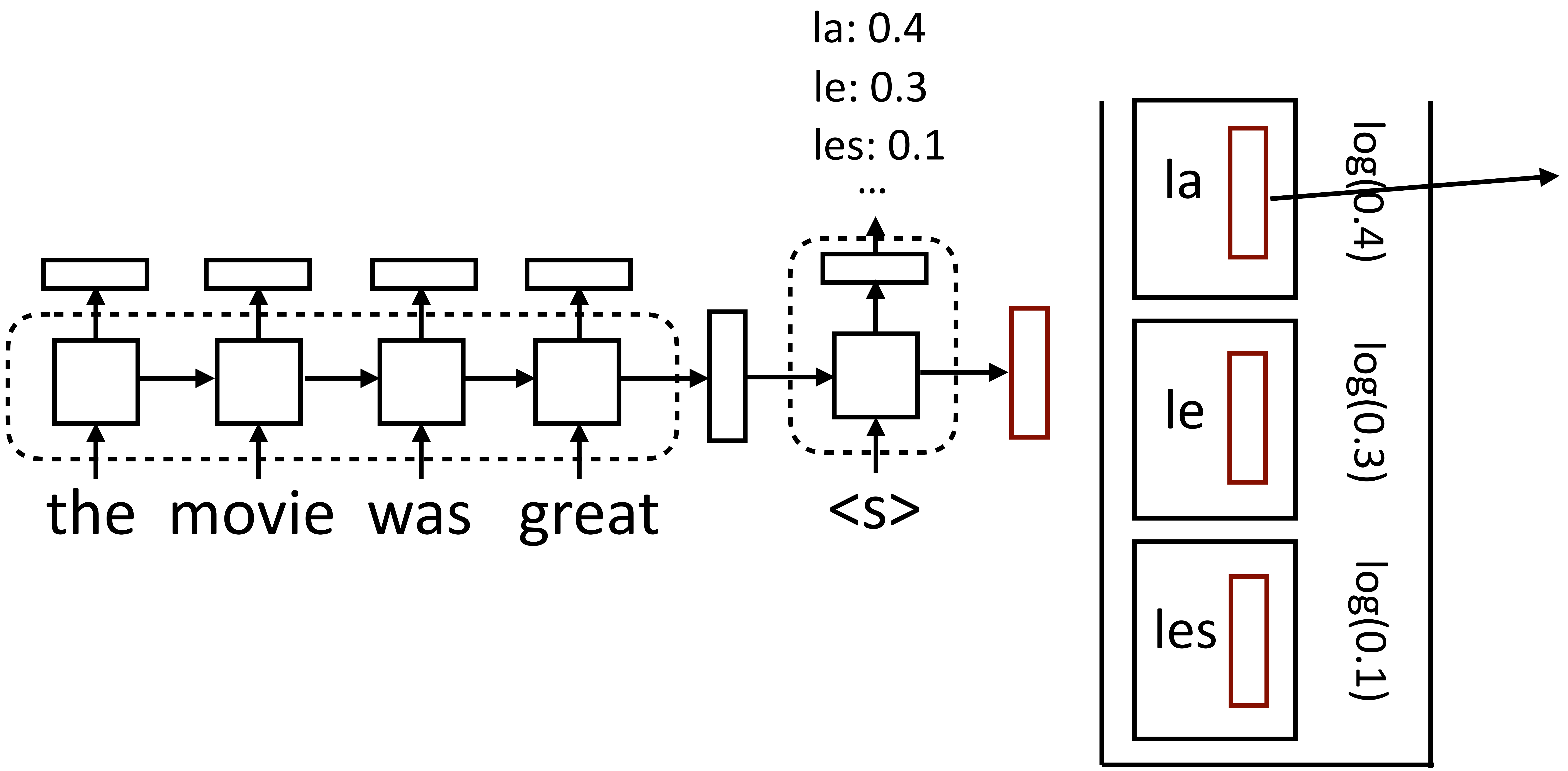
Beam Search

- Maintain decoder state, token history in beam



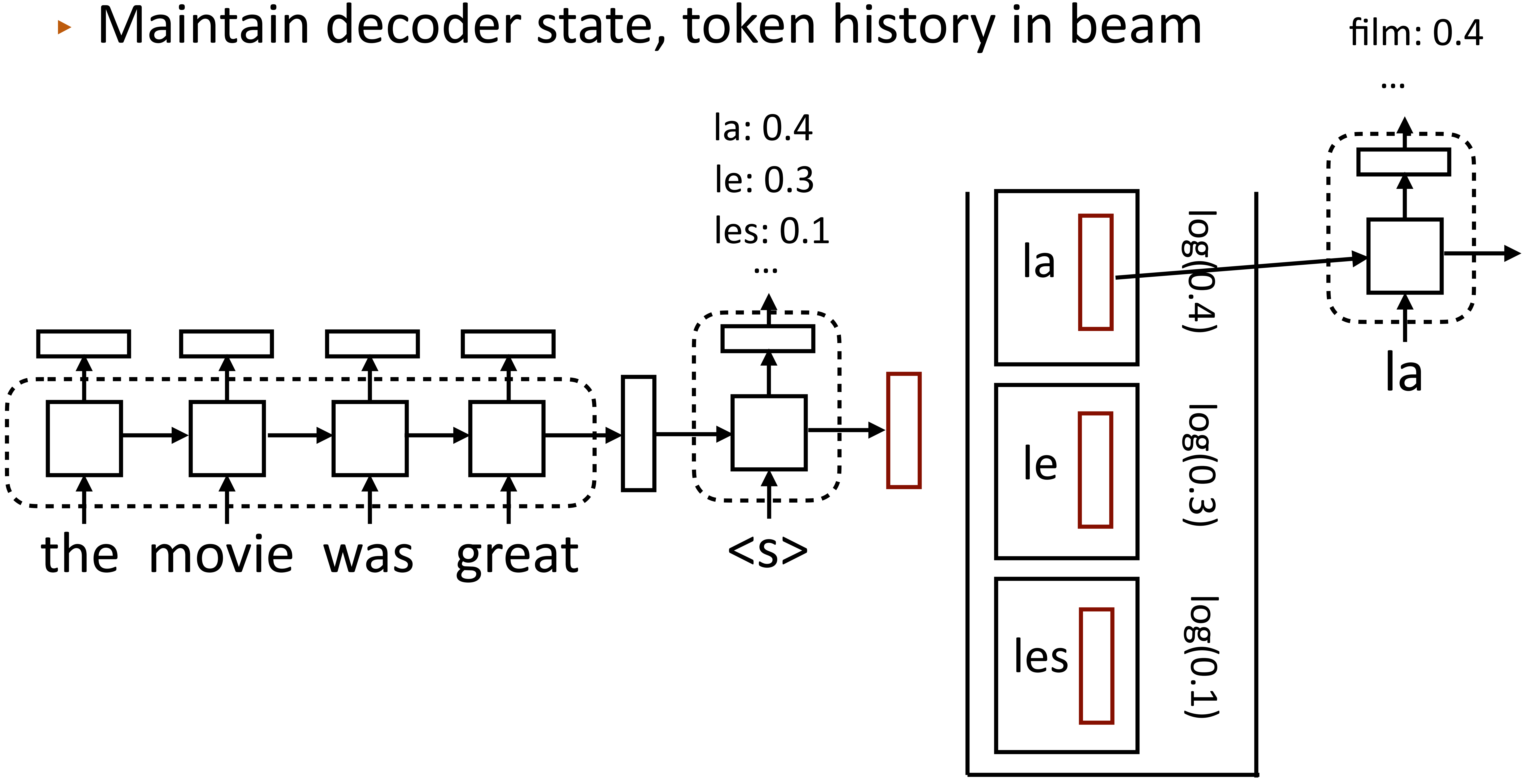
Beam Search

- Maintain decoder state, token history in beam



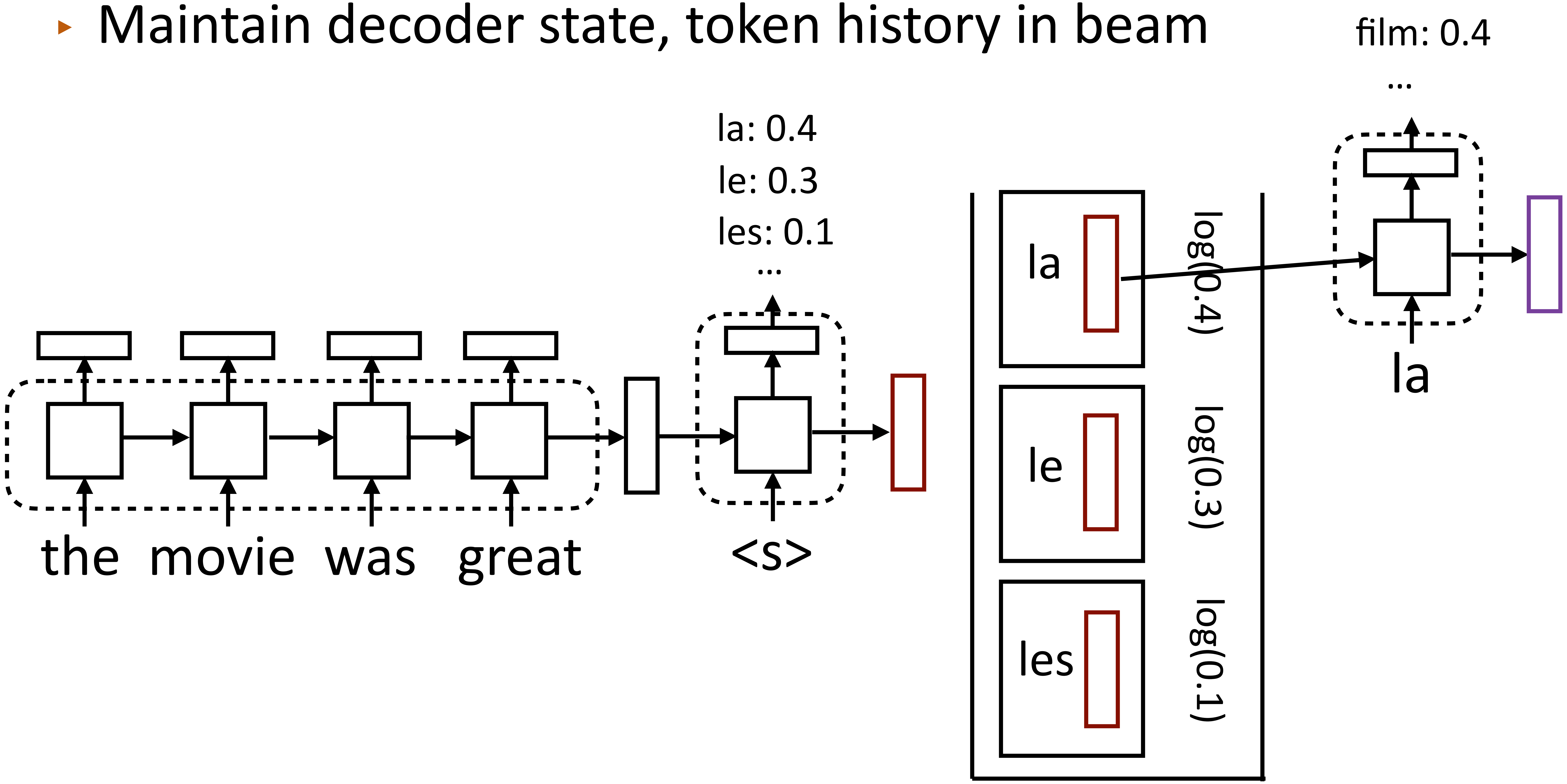
Beam Search

- Maintain decoder state, token history in beam



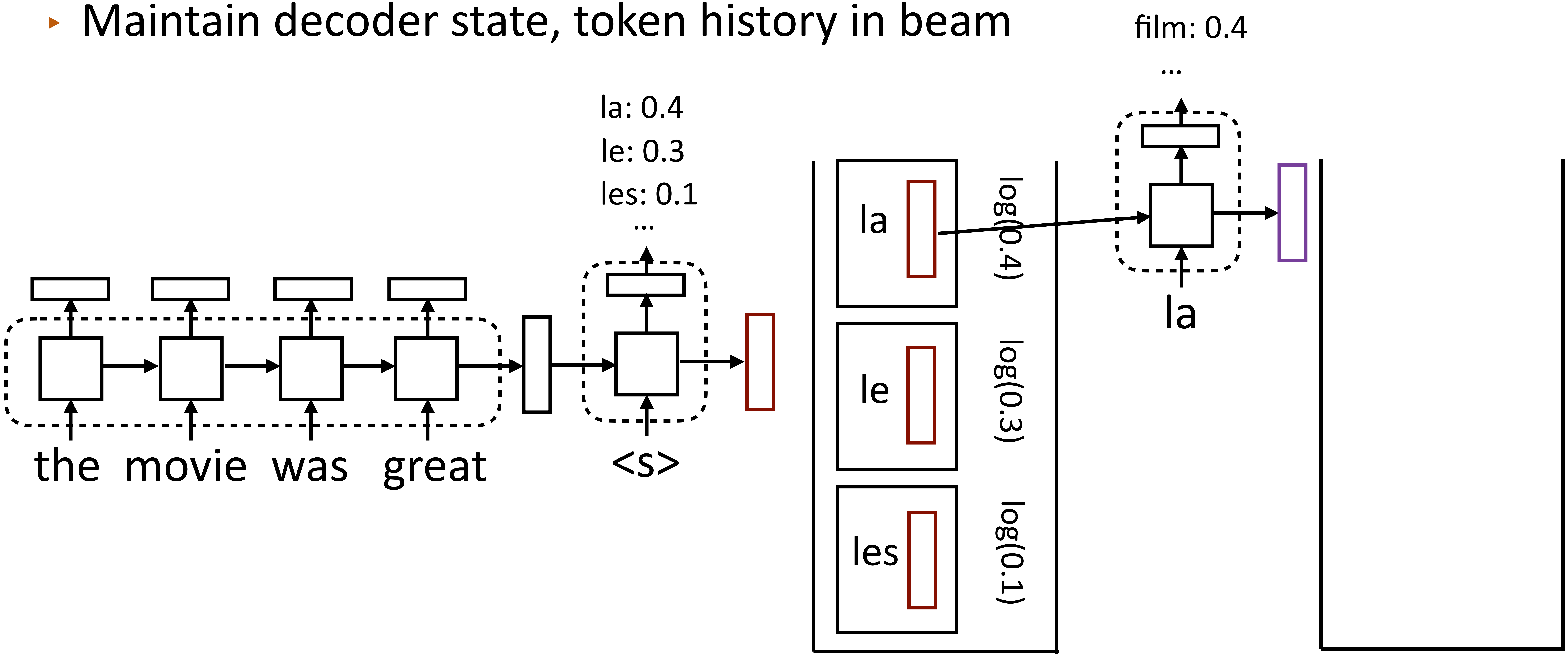
Beam Search

- Maintain decoder state, token history in beam



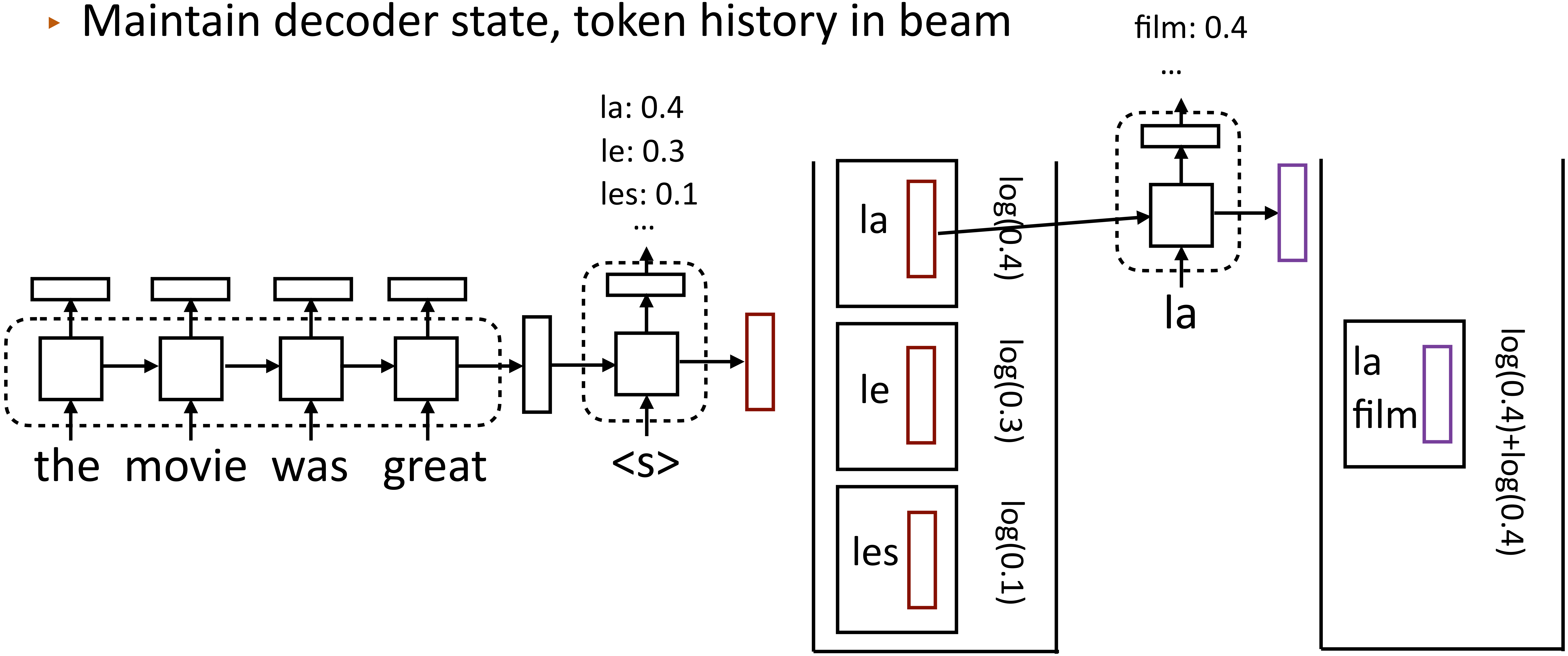
Beam Search

- Maintain decoder state, token history in beam



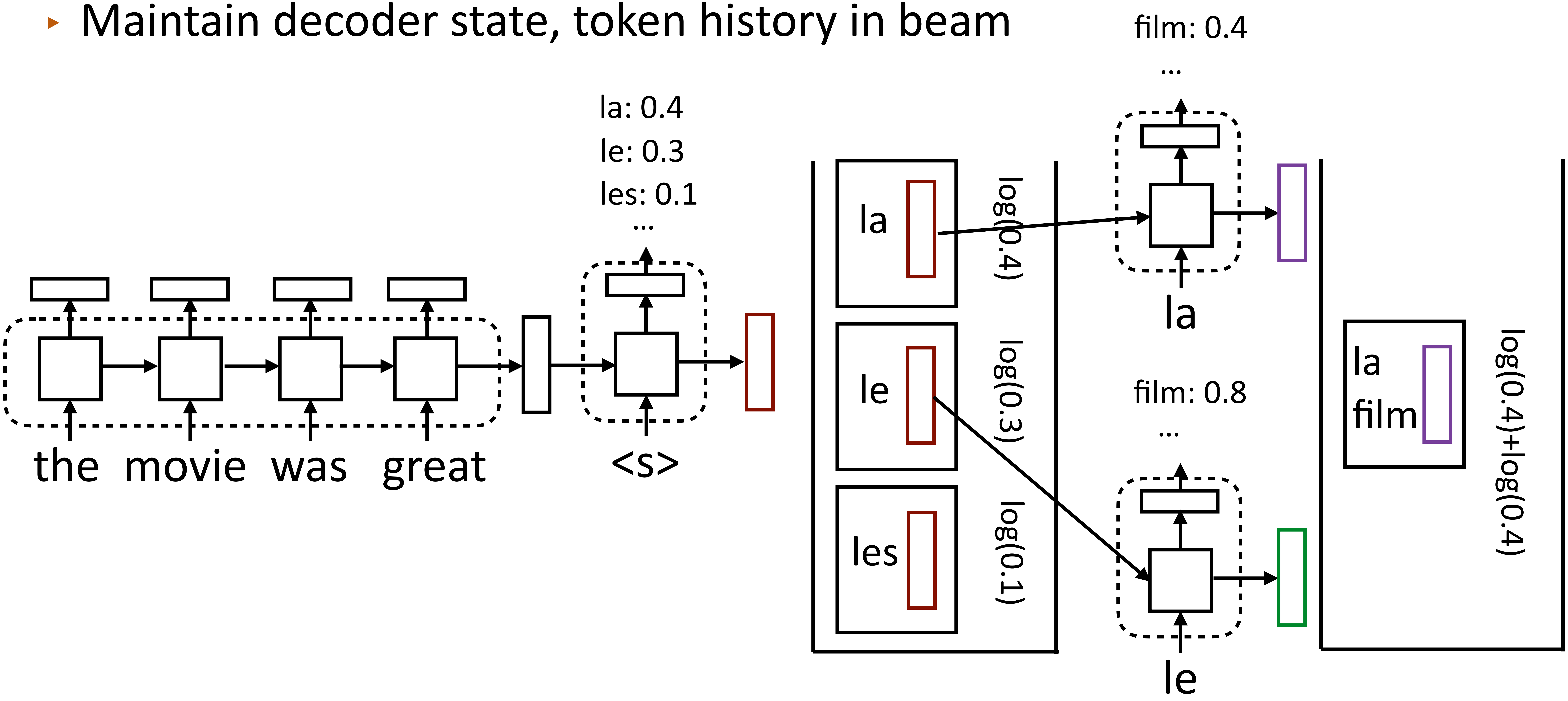
Beam Search

- Maintain decoder state, token history in beam



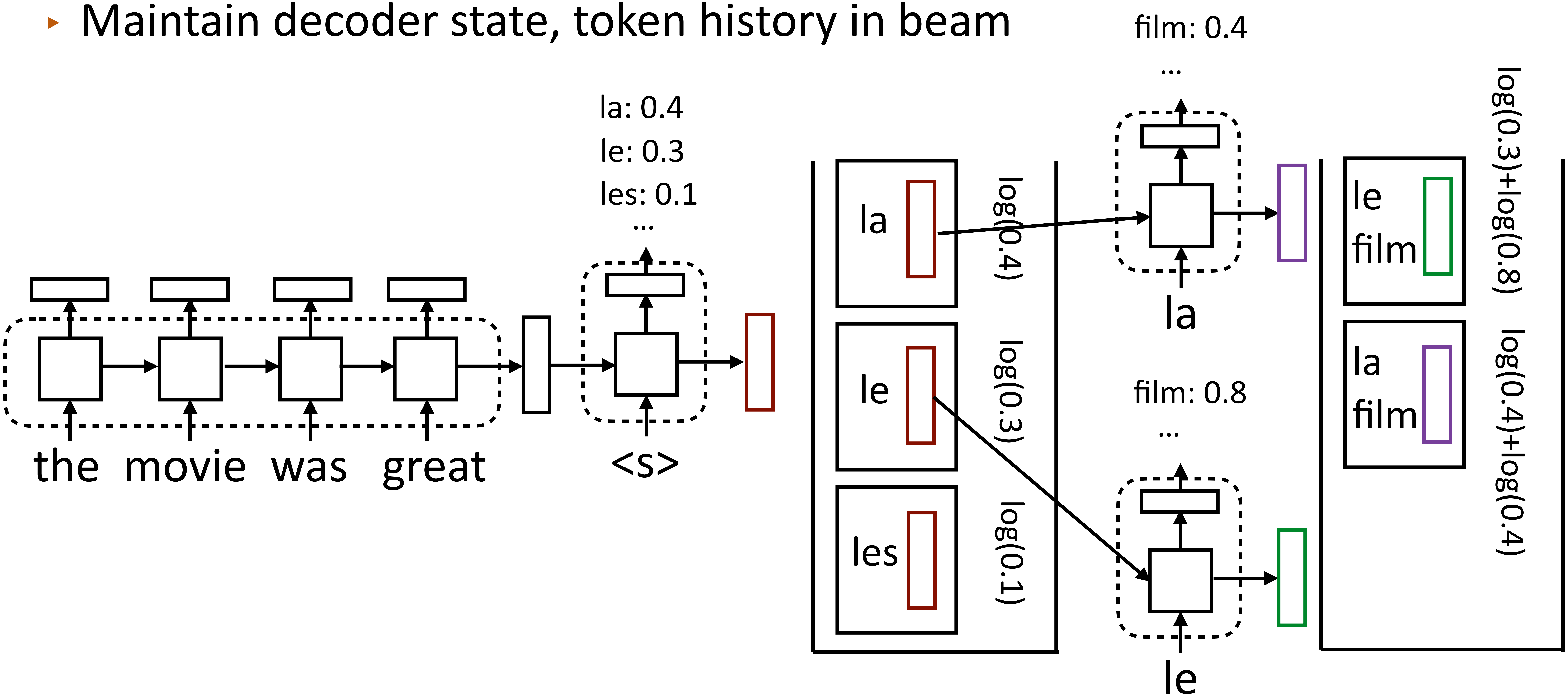
Beam Search

- Maintain decoder state, token history in beam



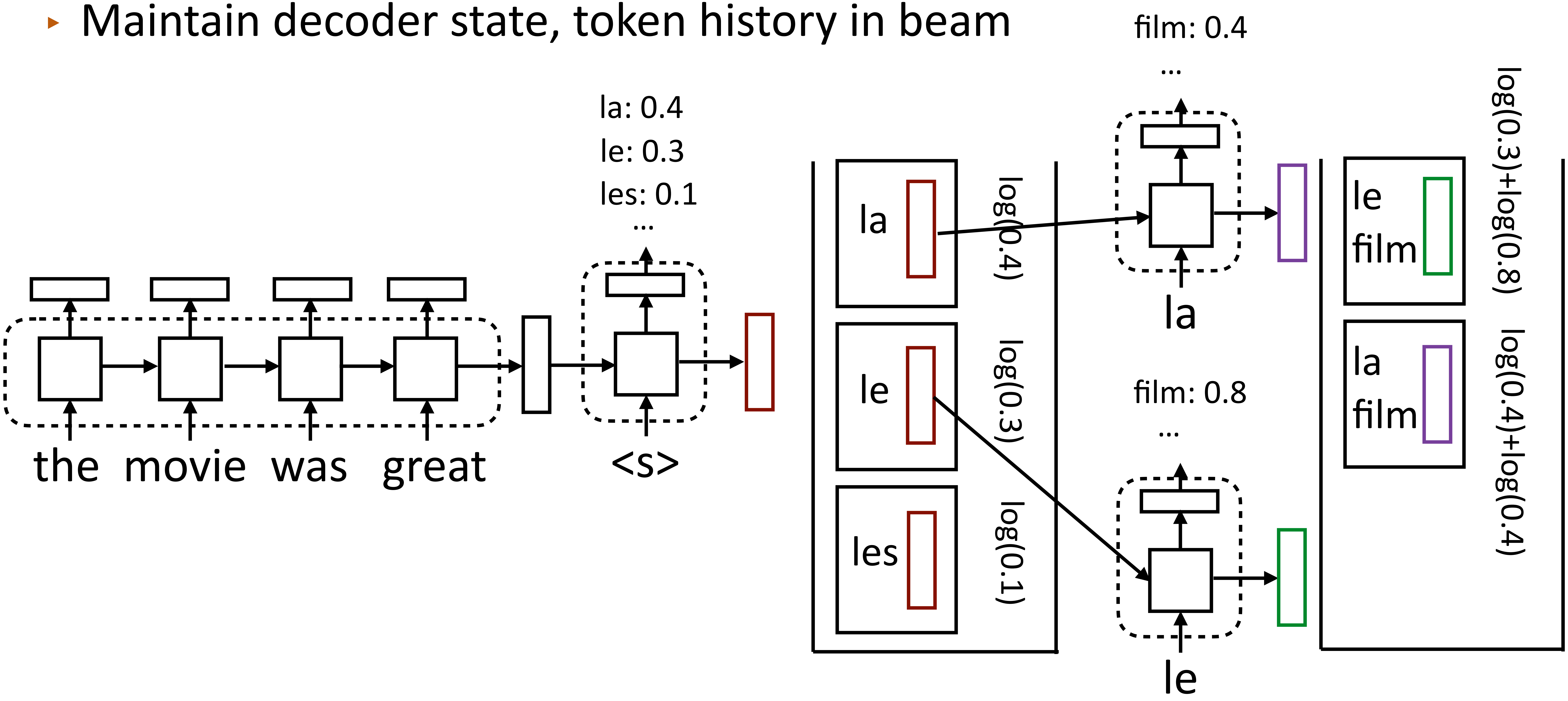
Beam Search

- Maintain decoder state, token history in beam



Beam Search

- Maintain decoder state, token history in beam



- Do **not** max over the two *film* states! Hidden state vectors are different

Regex Prediction

Locascio et al. (2016)

Regex Prediction

- Can use for other translation-like tasks

Locascio et al. (2016)

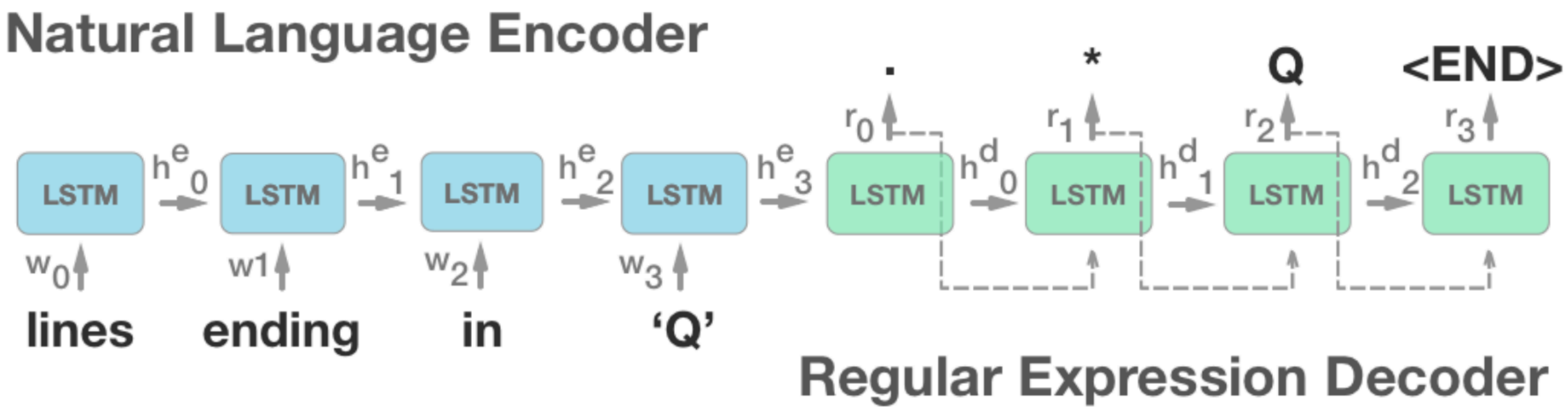
Regex Prediction

- Can use for other translation-like tasks
- Predict regex from text

Locascio et al. (2016)

Regex Prediction

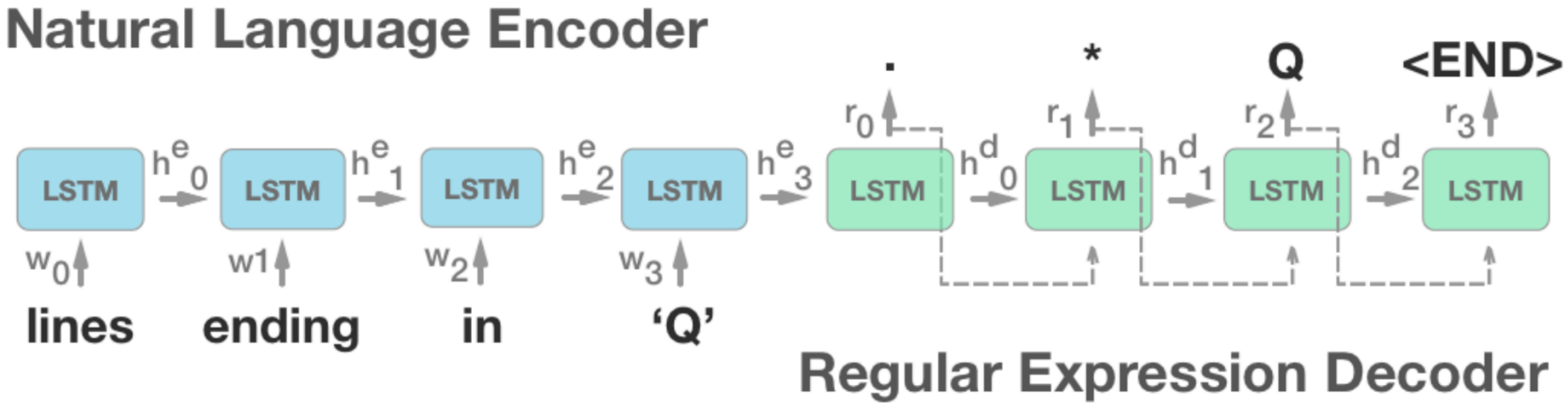
- Can use for other translation-like tasks
- Predict regex from text



Locascio et al. (2016)

Regex Prediction

- Can use for other translation-like tasks
- Predict regex from text



- Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes

Locascio et al. (2016)

SQL Generation

- Convert natural language description into a SQL query against some DB

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = “York”
```

Zhong et al. (2017)

SQL Generation

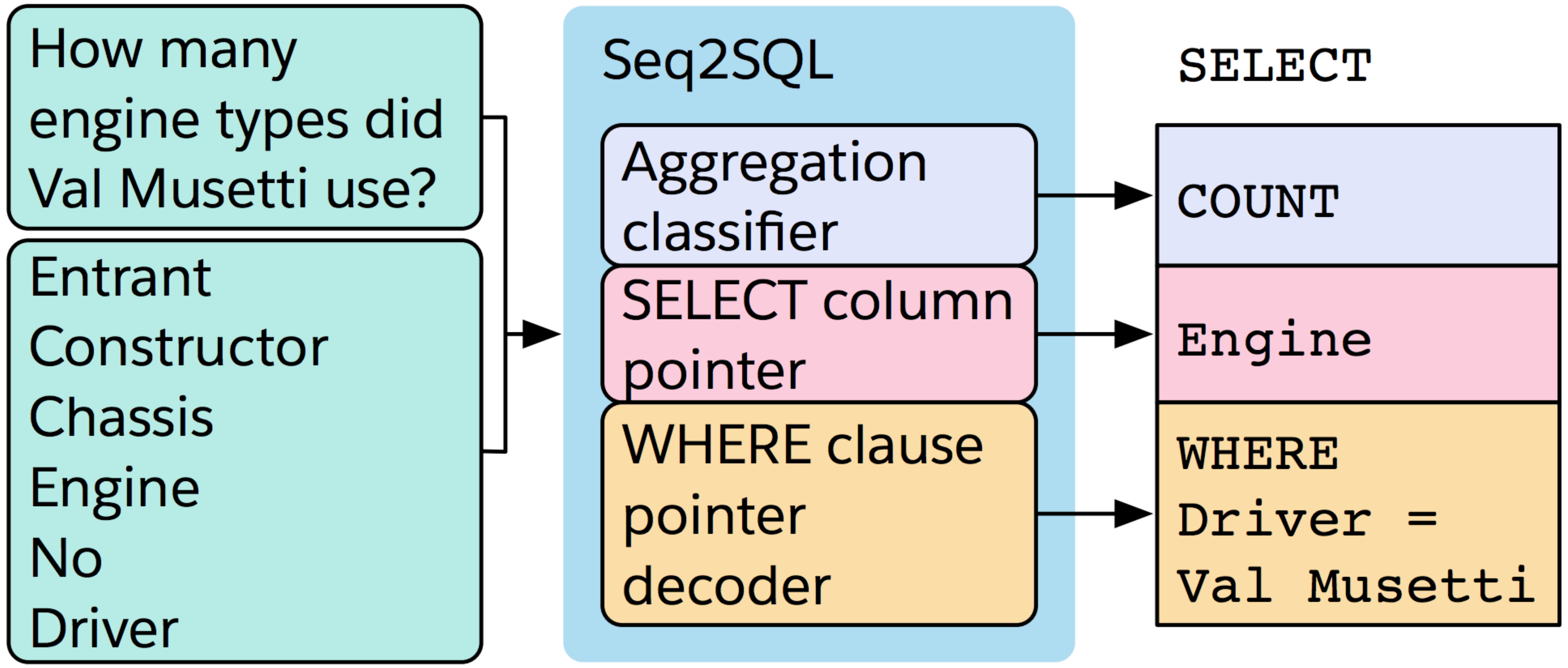
- Convert natural language description into a SQL query against some DB

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = "York"
```



Zhong et al. (2017)

SQL Generation

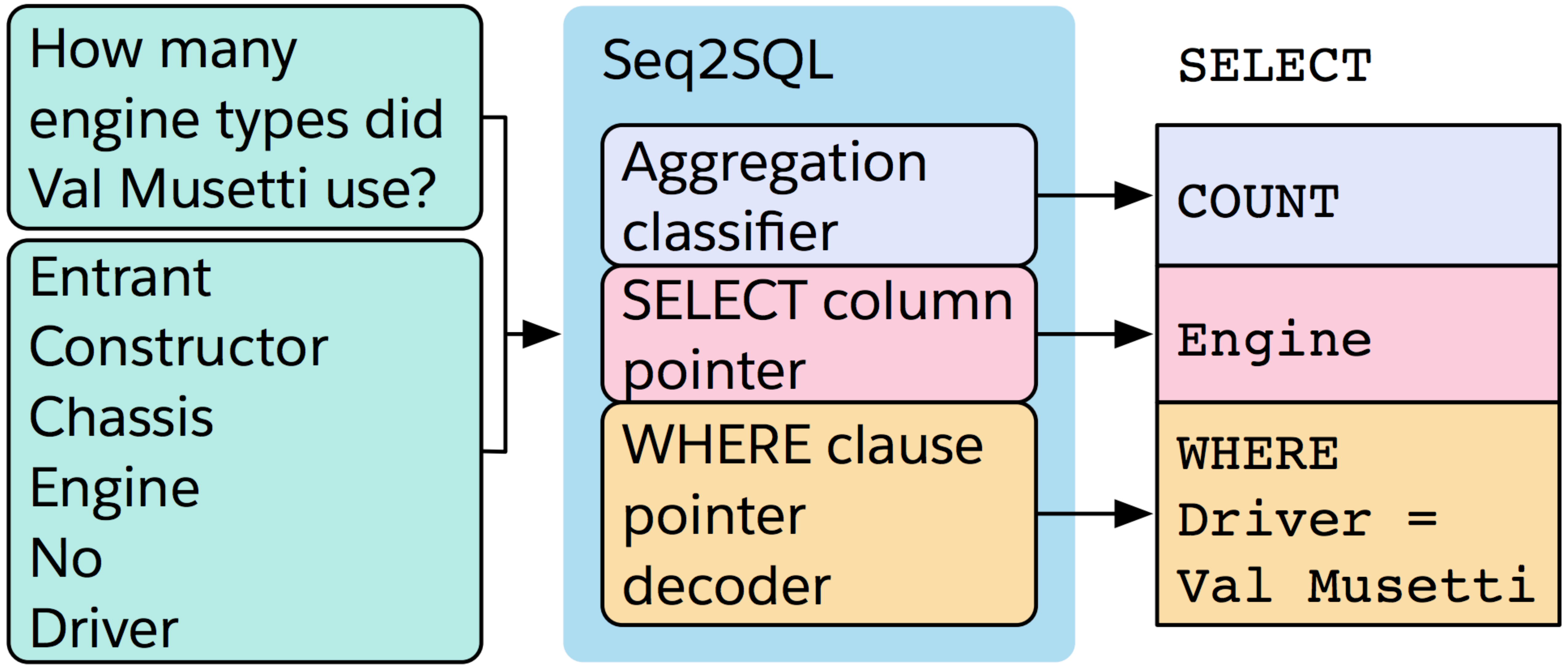
- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = "York"
```



Zhong et al. (2017)

SQL Generation

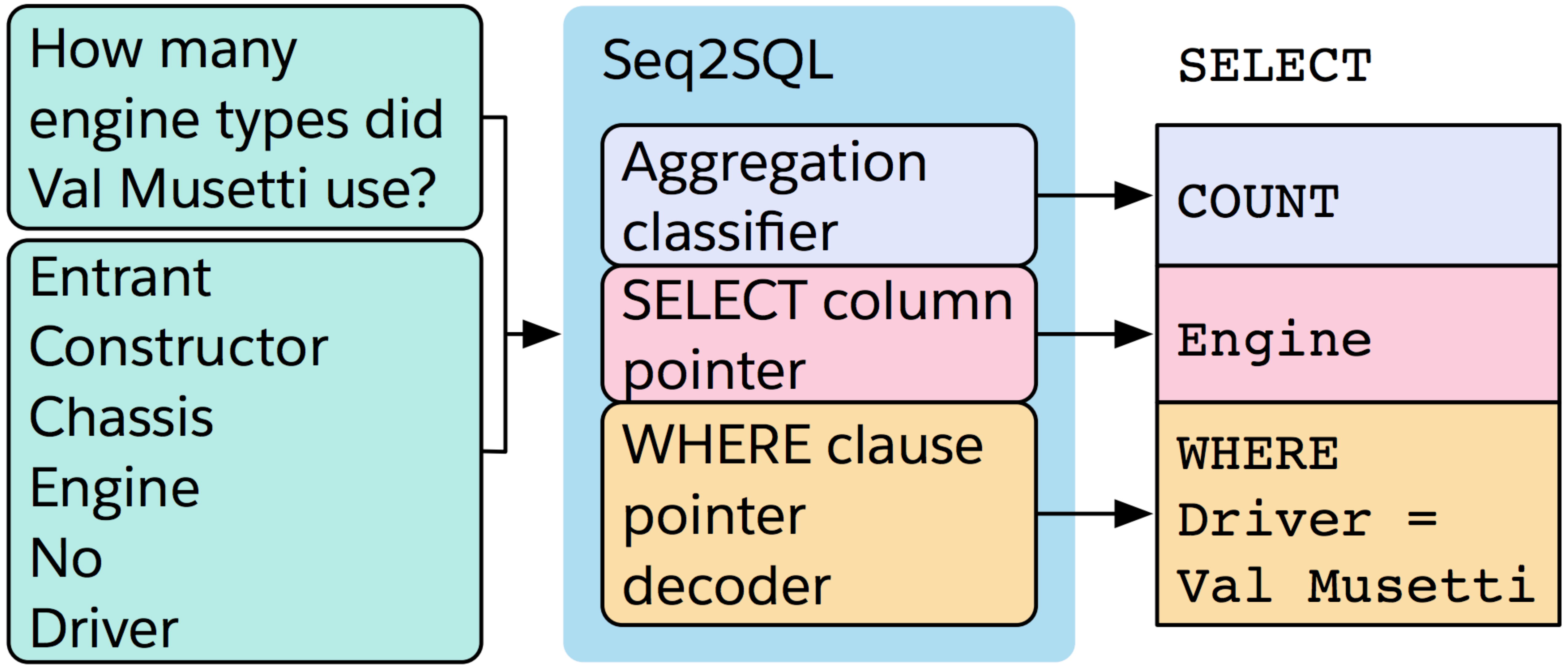
- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?
  - Three seq2seq models

Question:

How many CFL teams are from York College?

SQL:

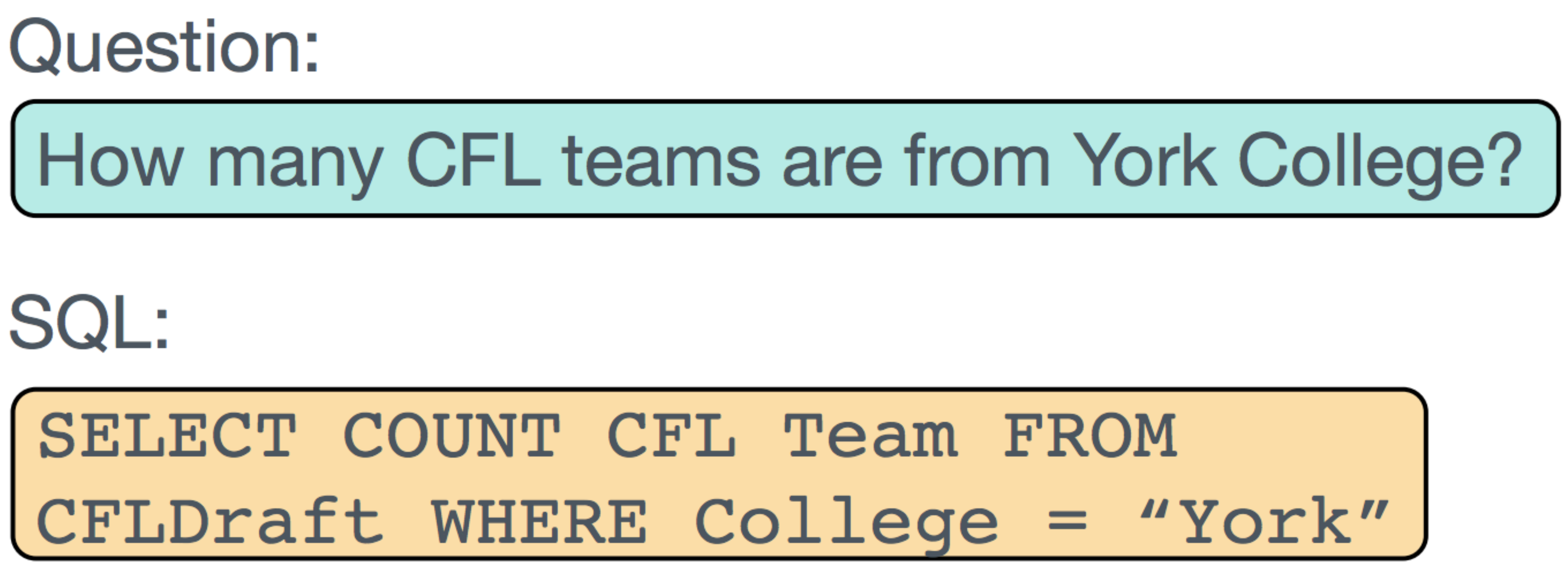
```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = “York”
```

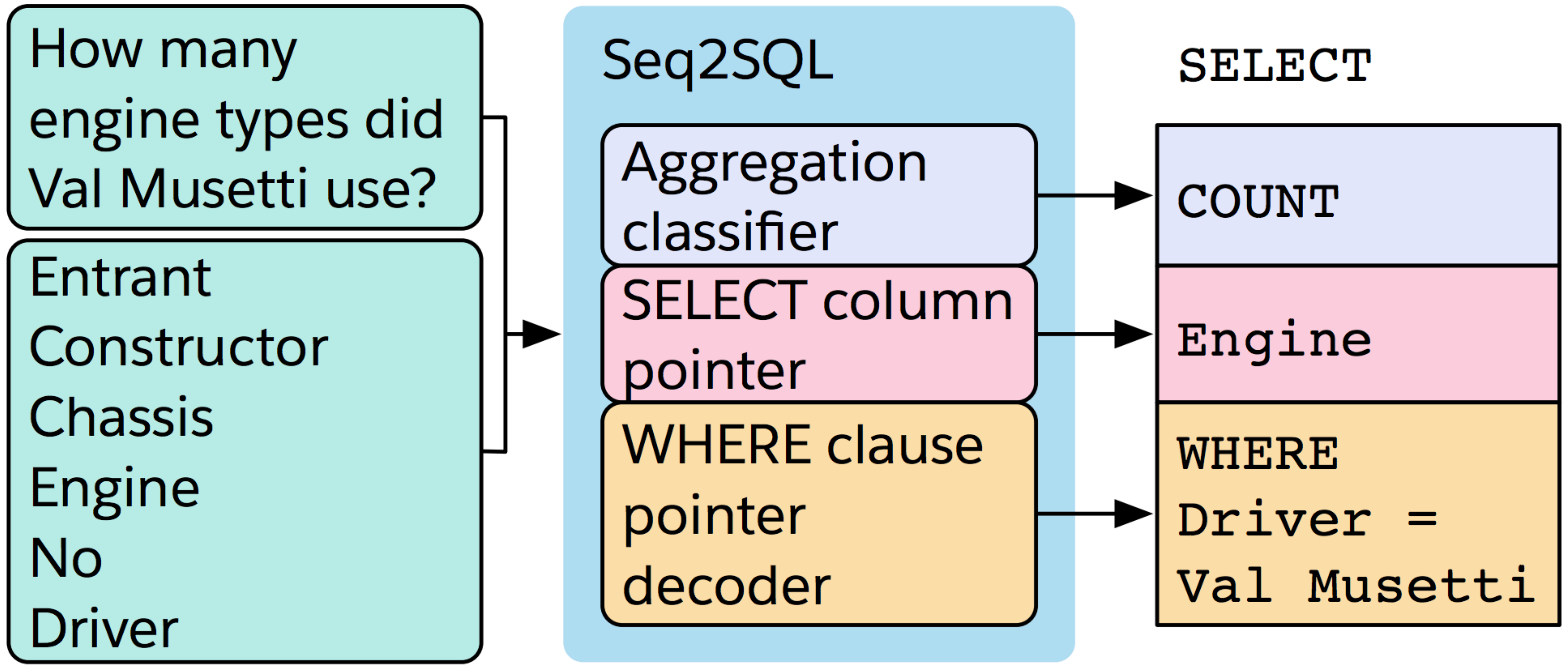


Zhong et al. (2017)

SQL Generation

- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?
  - Three seq2seq models
- How to capture column names + constants?





Zhong et al. (2017)

SQL Generation

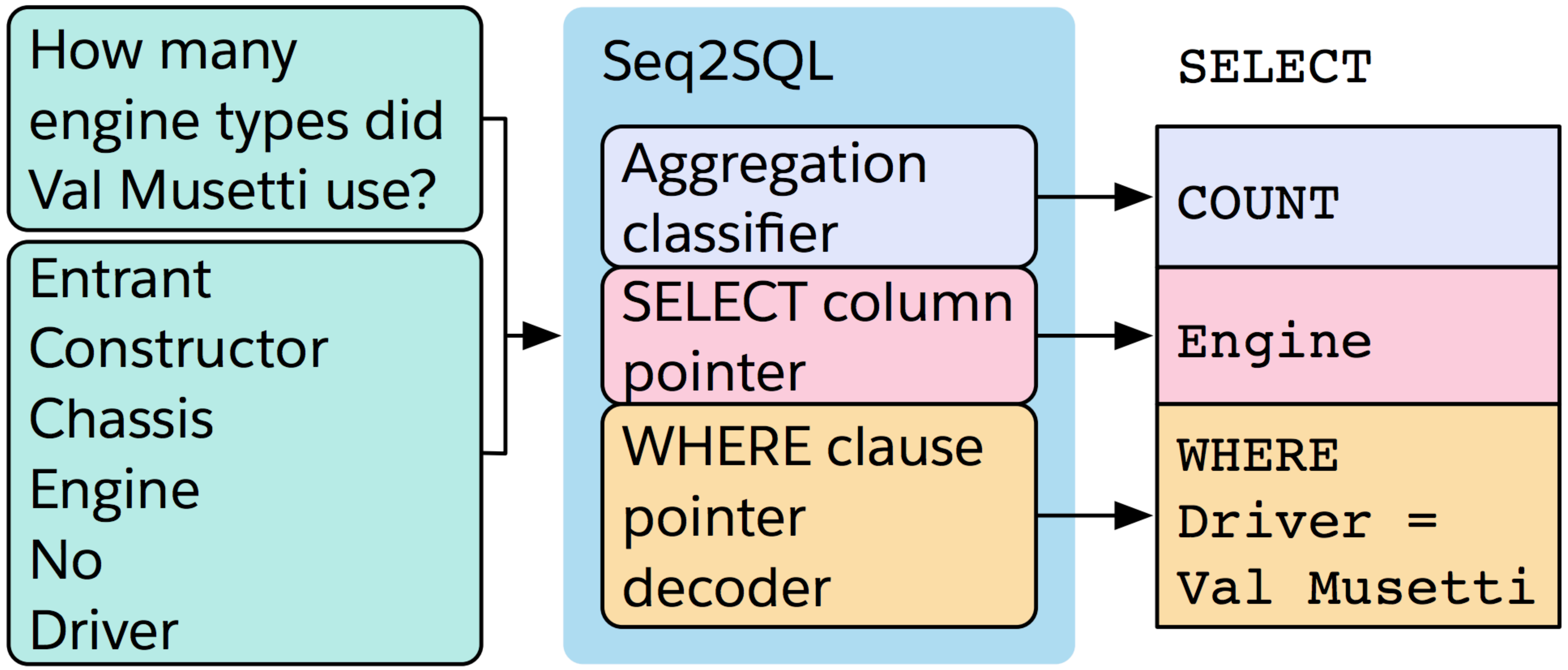
- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?
  - Three seq2seq models
- How to capture column names + constants?
  - Pointer mechanisms

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = “York”
```



Zhong et al. (2017)

Attention

Problems with Seq2seq Models

- Encoder-decoder models like to repeat themselves:

Problems with Seq2seq Models

- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige → A boy plays in the snow **boy plays boy plays**

Problems with Seq2seq Models

- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige $\rightarrow$ A boy plays in the snow **boy plays boy plays**

- Often a byproduct of training these models poorly

Problems with Seq2seq Models

- Encoder-decoder models like to repeat themselves:

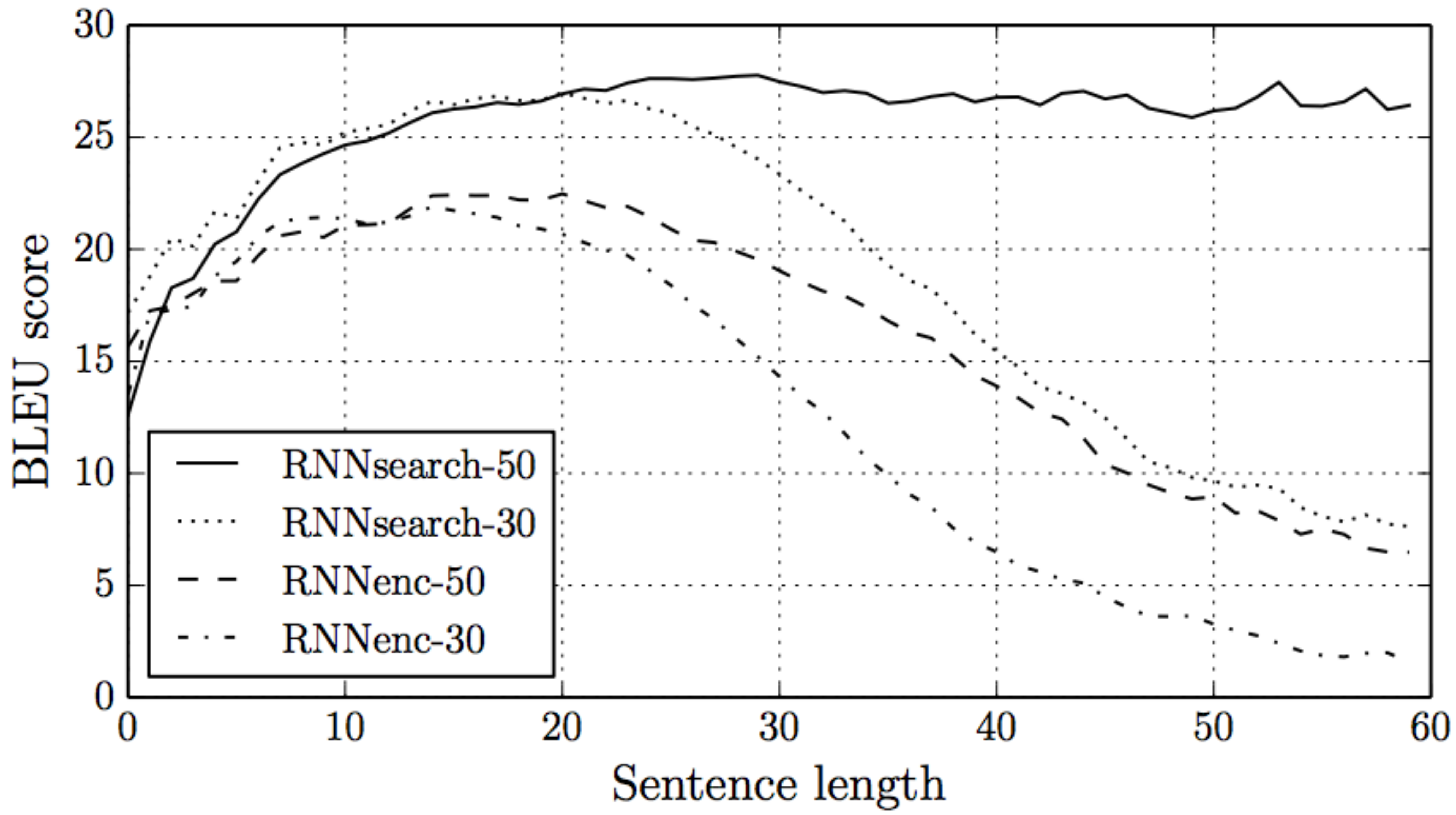
Un garçon joue dans la neige → A boy plays in the snow **boy plays boy plays**

- Often a byproduct of training these models poorly

- Need some notion of input coverage or what input words we've translated

Problems with Seq2seq Models

- Bad at long sentences: 1) a fixed-size representation doesn't scale; 2) LSTMs still have a hard time remembering for really long periods of time

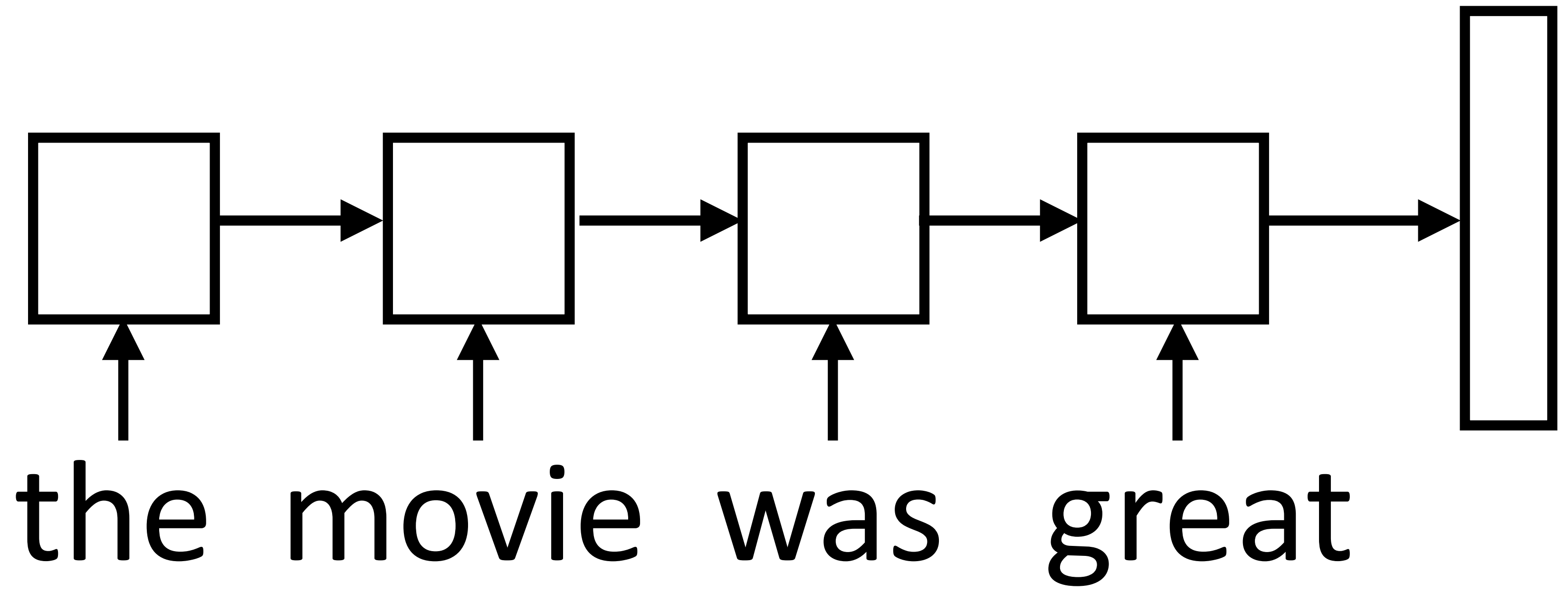


RNNsearch: introduces attention mechanism to give "variable-sized" representation

Bahdanau et al. (2014)

Encoder-Decoder

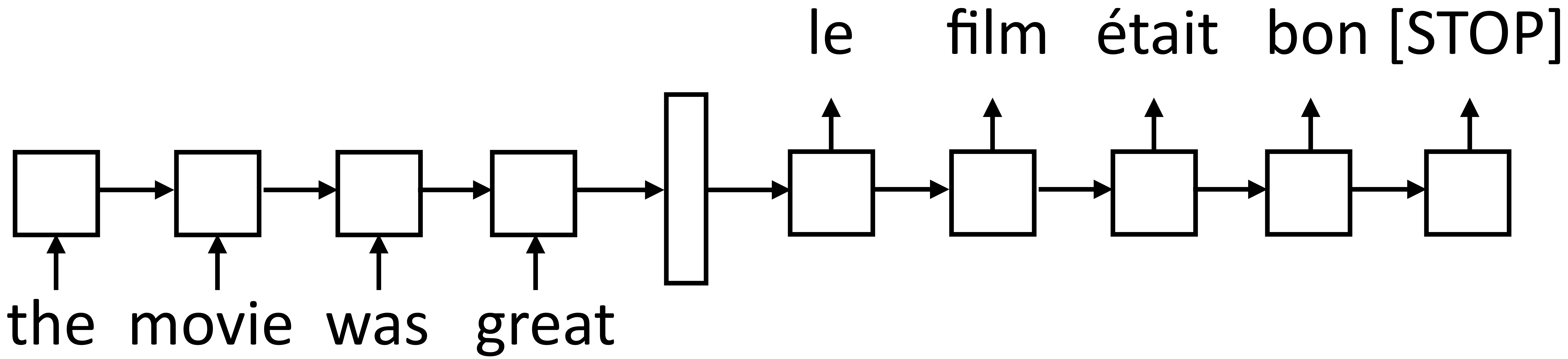
- Encode a sequence into a fixed-sized vector



Sutskever et al. (2014)

Encoder-Decoder

- Encode a sequence into a fixed-sized vector



- Now use that vector to produce a series of tokens as output from a separate LSTM *decoder*

Sutskever et al. (2014)

Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

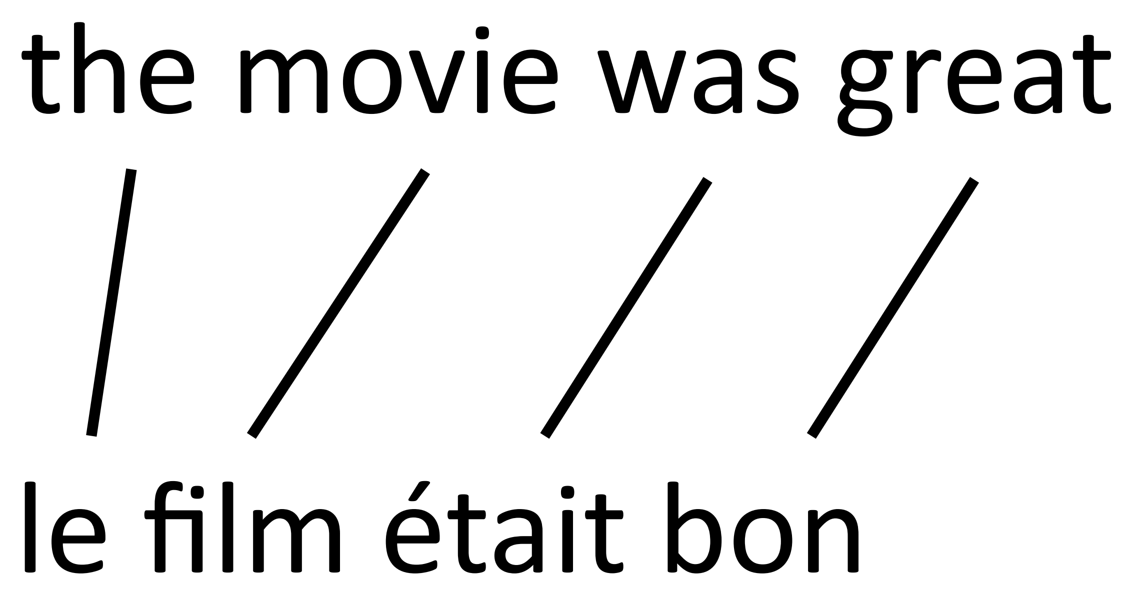
the movie was great

le film était bon

Aligned Inputs

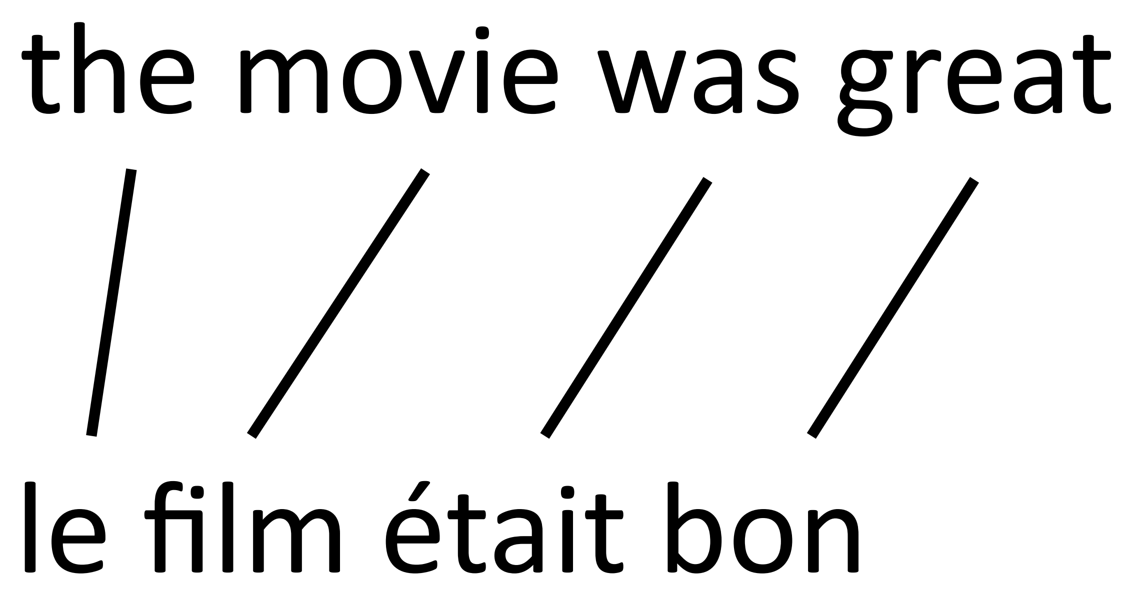
- Suppose we knew the source and target would be word-by-word translated

- Can look at the corresponding input word when translating — this could scale!

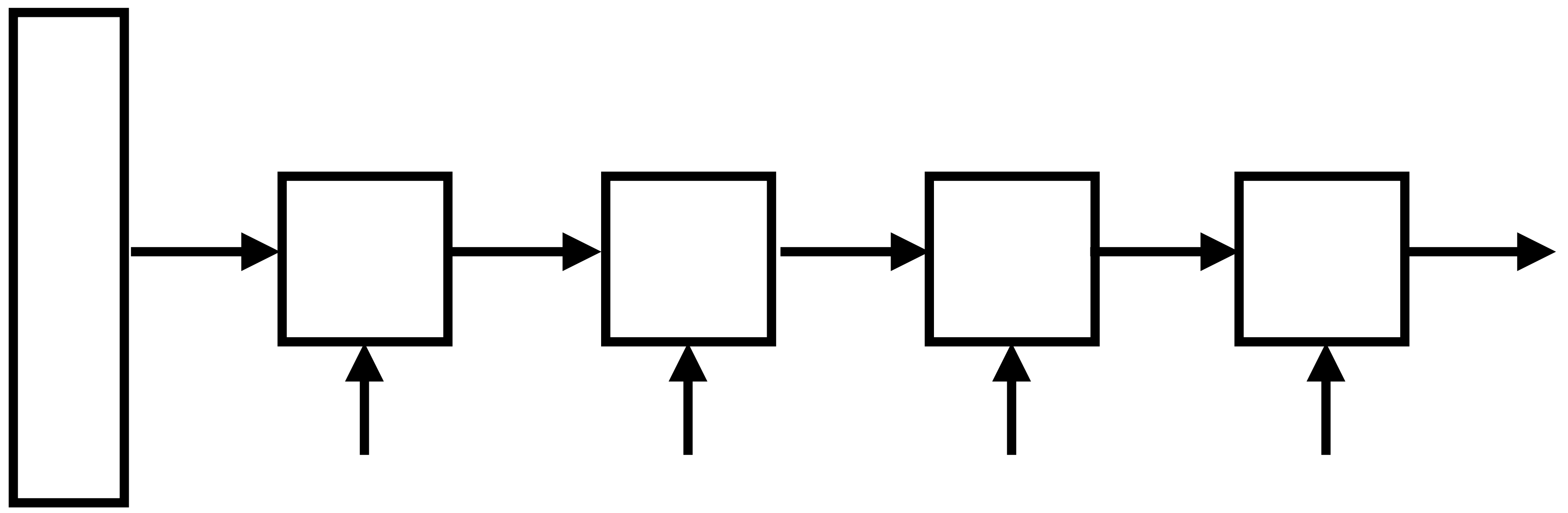


Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

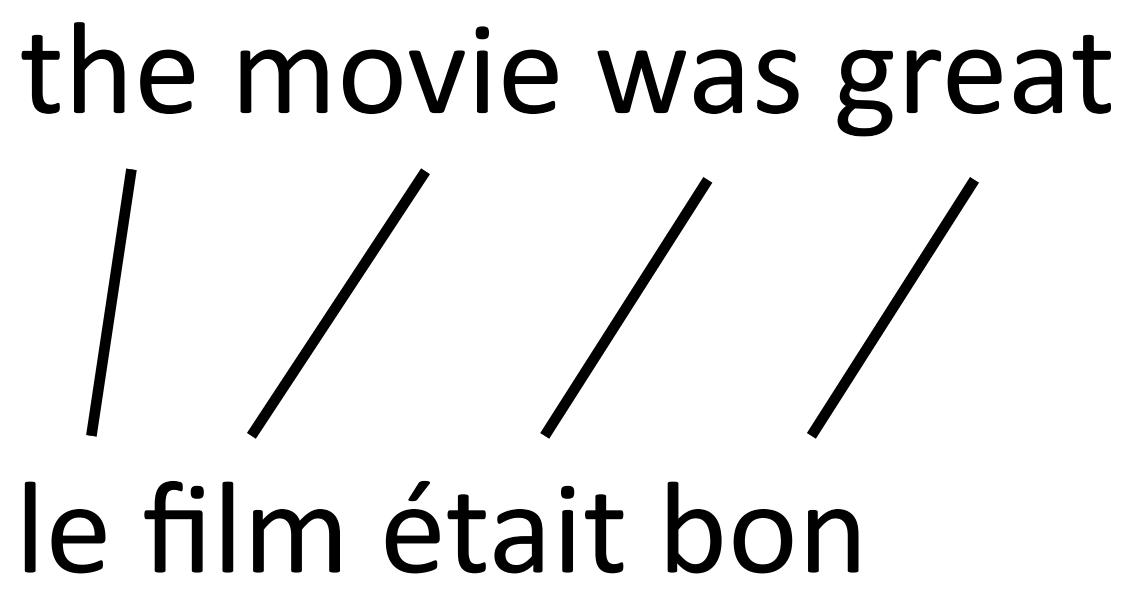


- Can look at the corresponding input word when translating — this could scale!

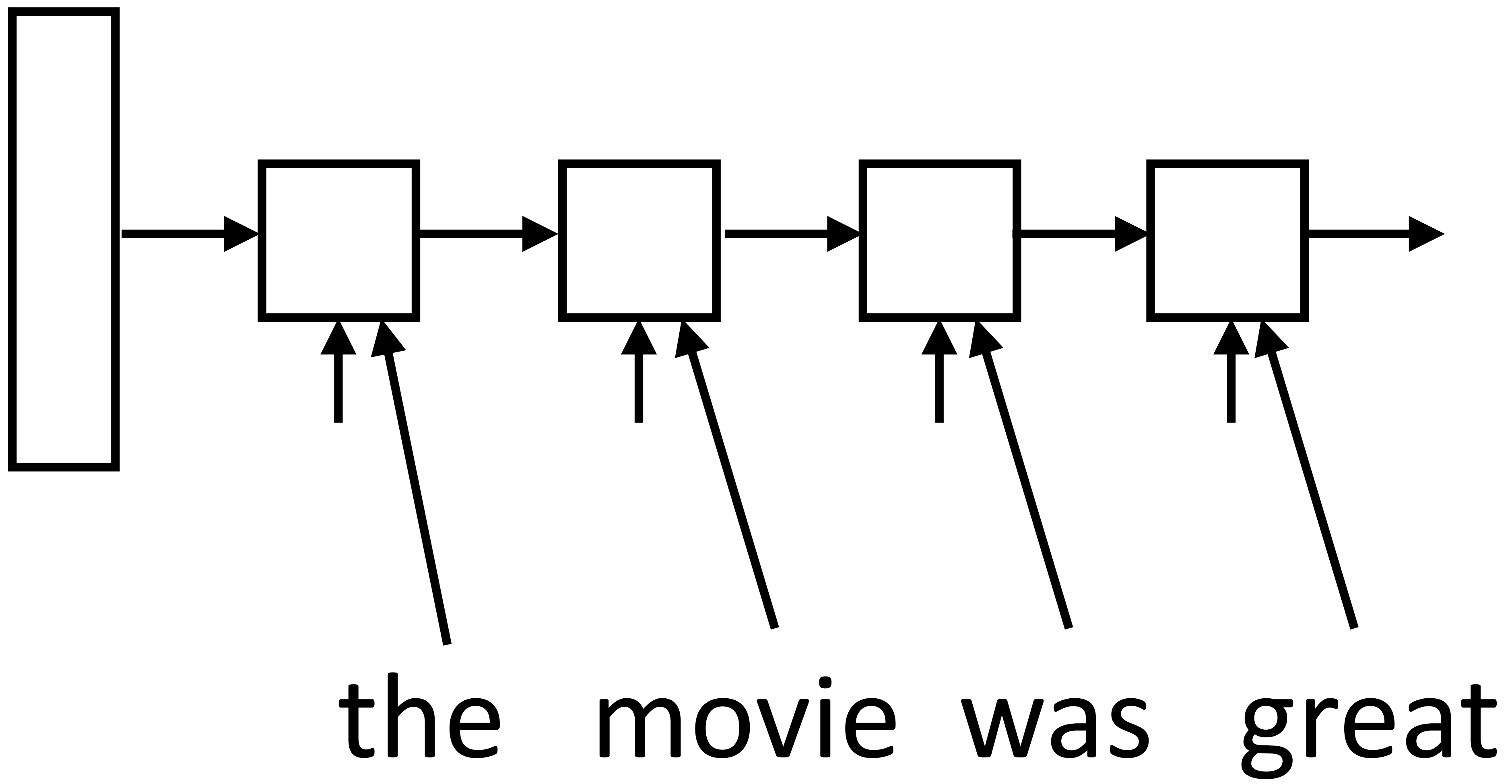


Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

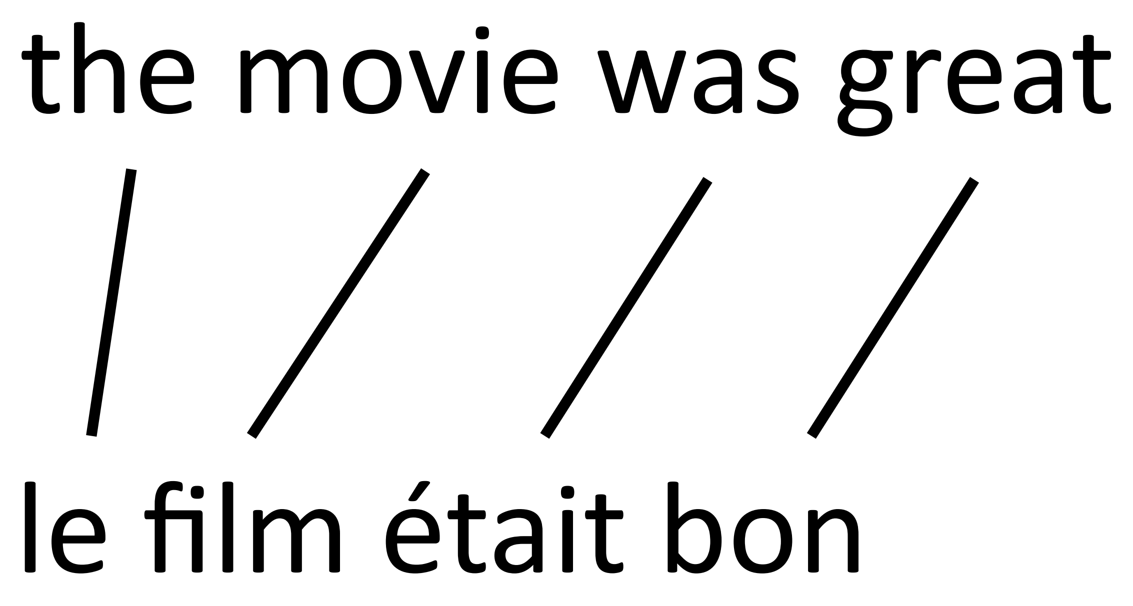


- Can look at the corresponding input word when translating — this could scale!

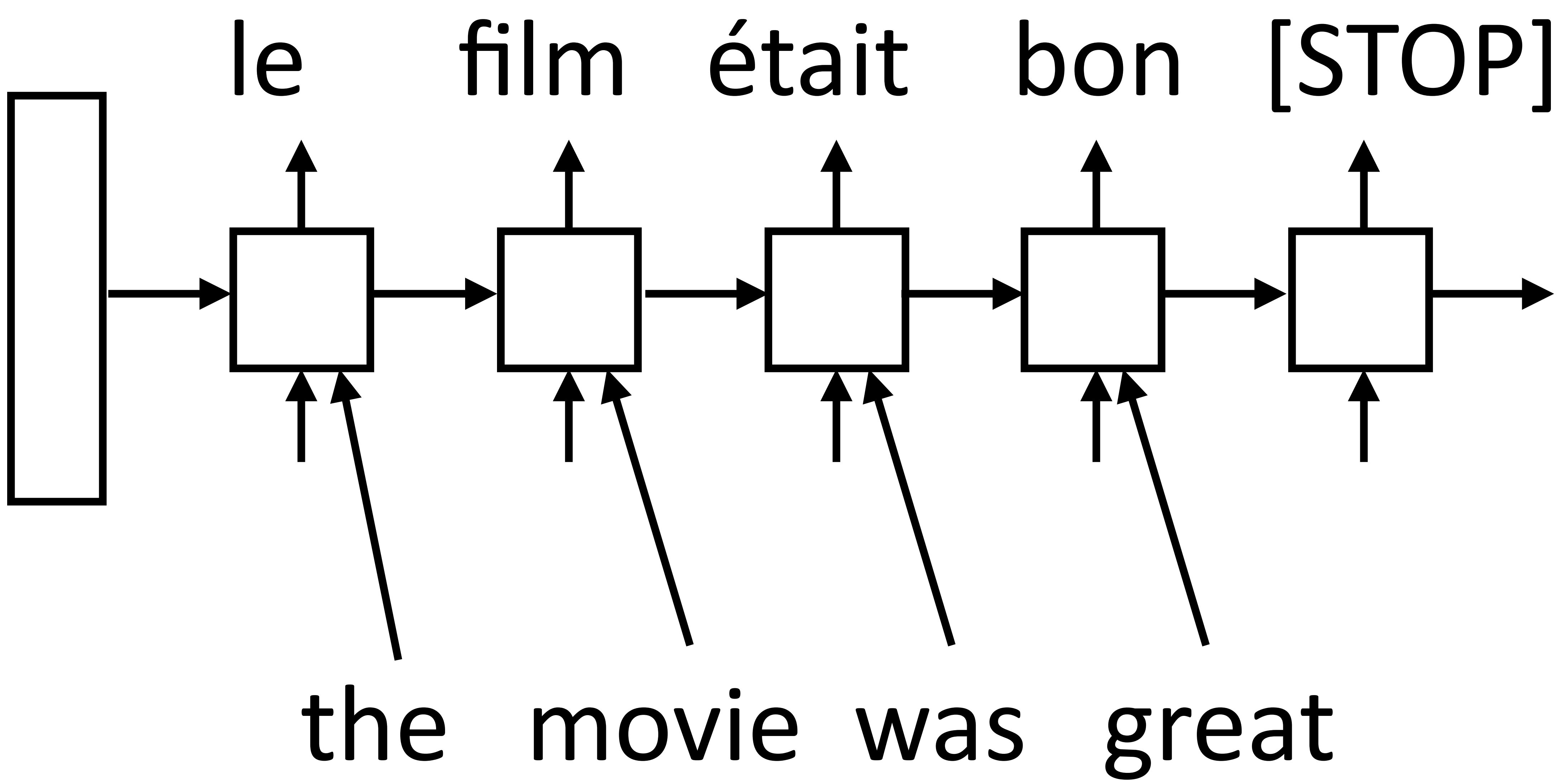


Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

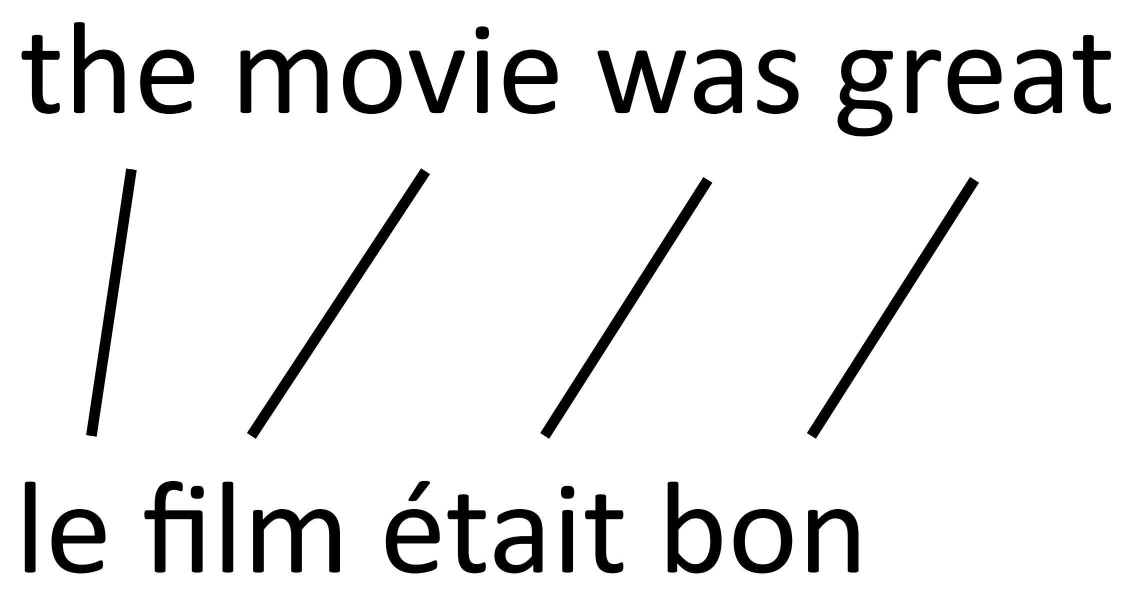


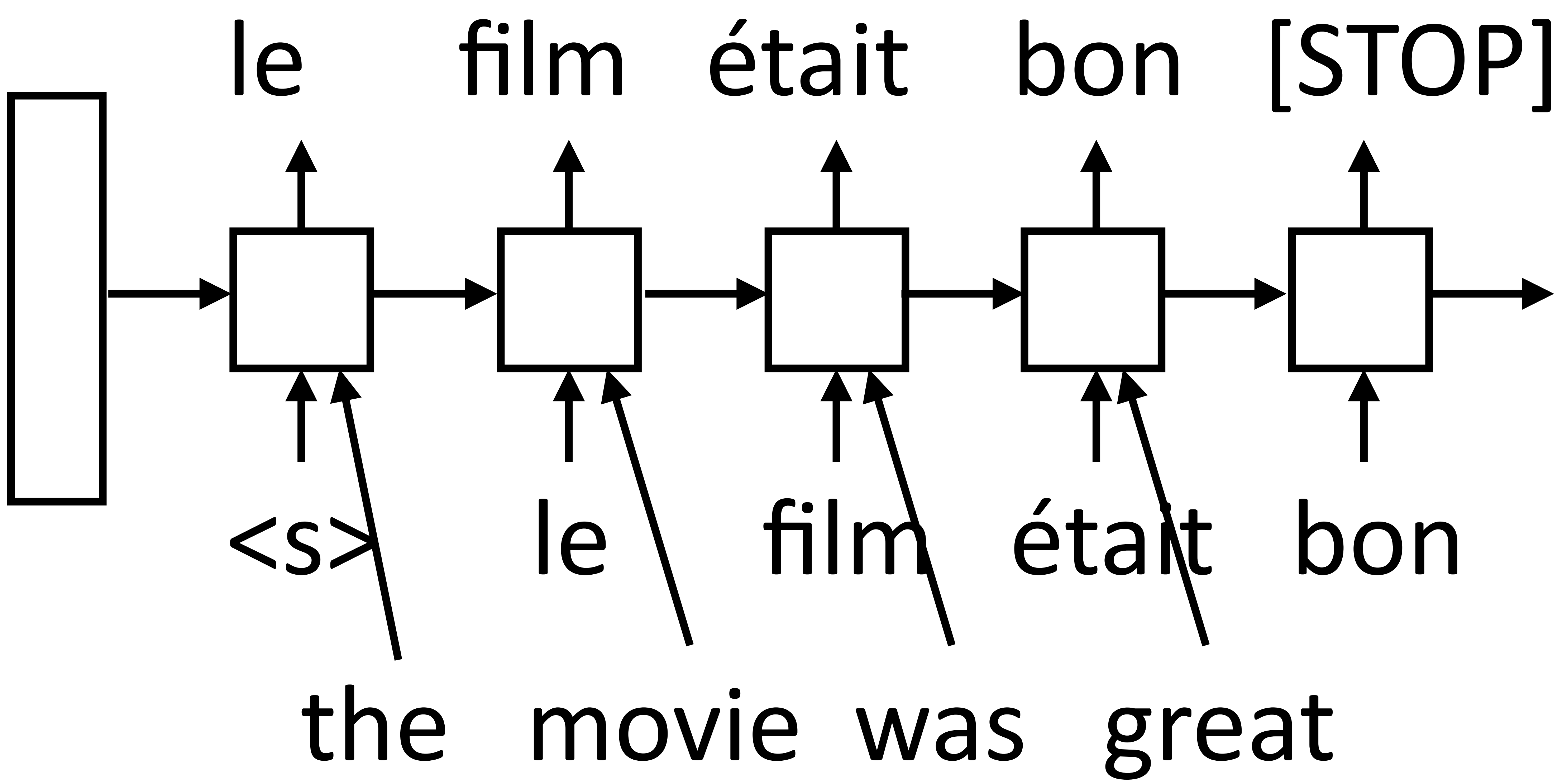
- Can look at the corresponding input word when translating — this could scale!



Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated
- Can look at the corresponding input word when translating — this could scale!



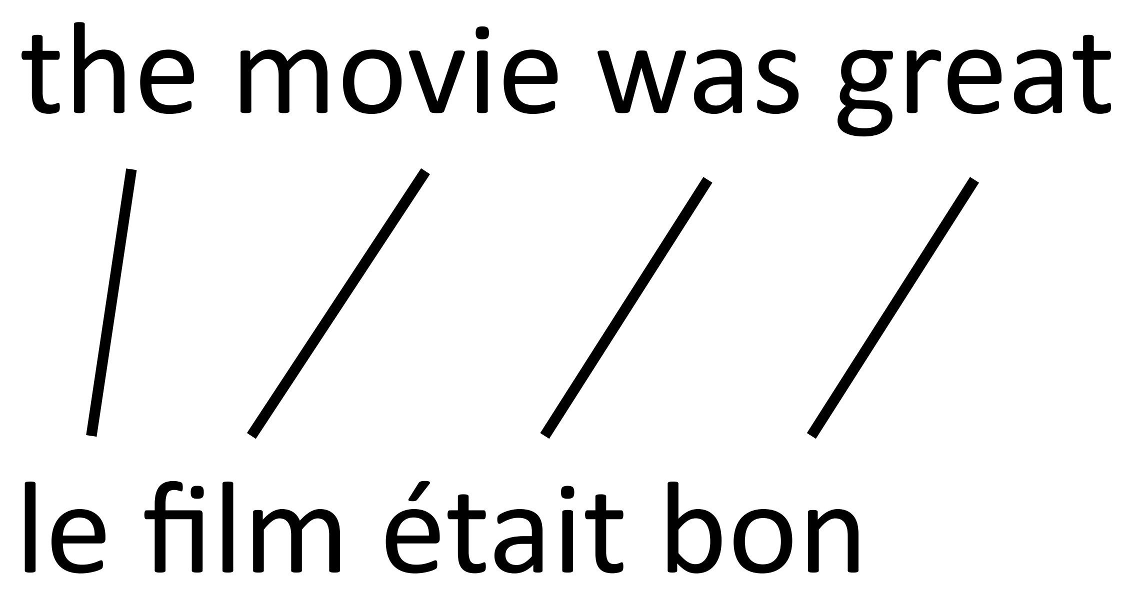


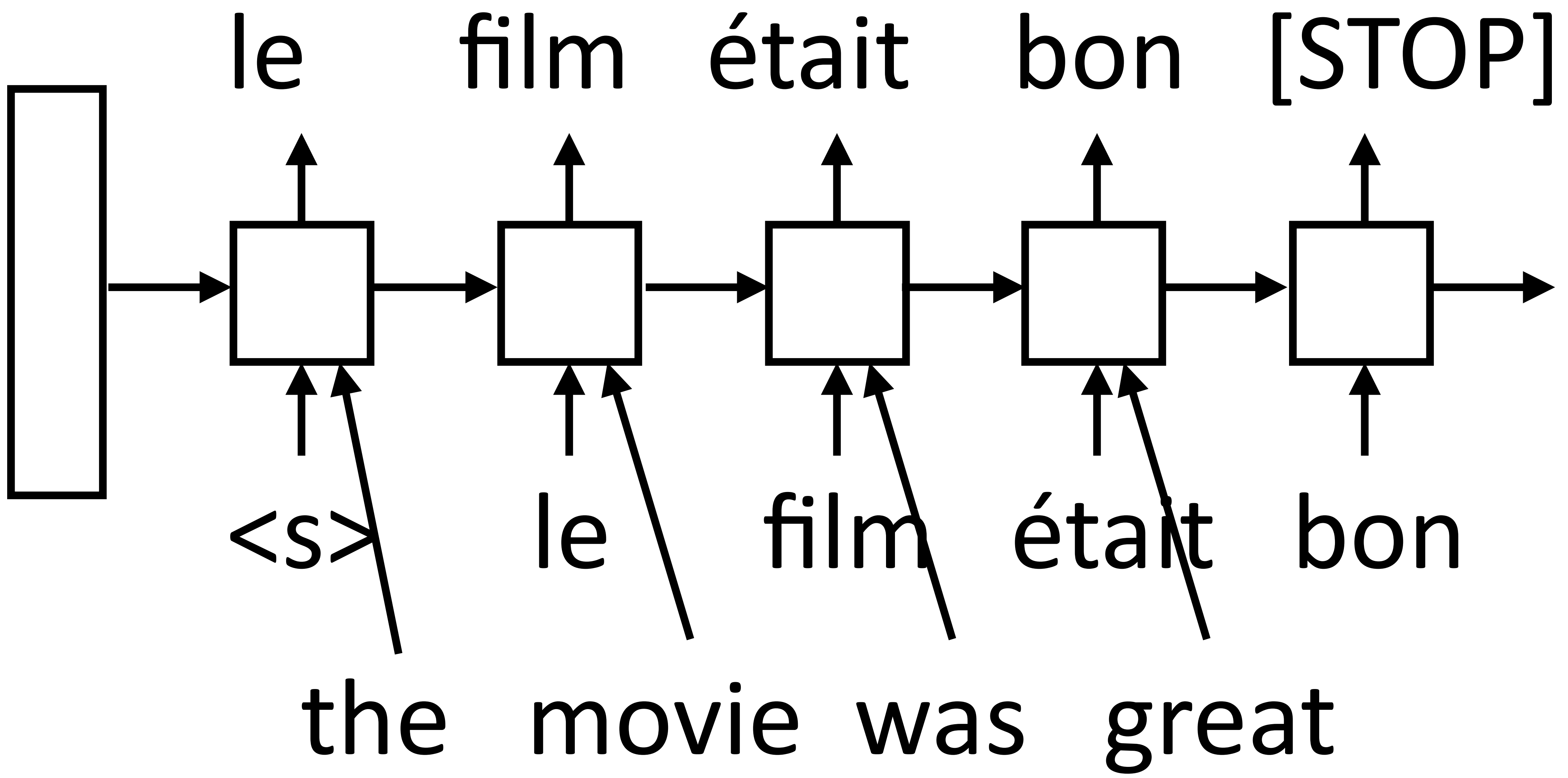
Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

- Can look at the corresponding input word when translating — this could scale!

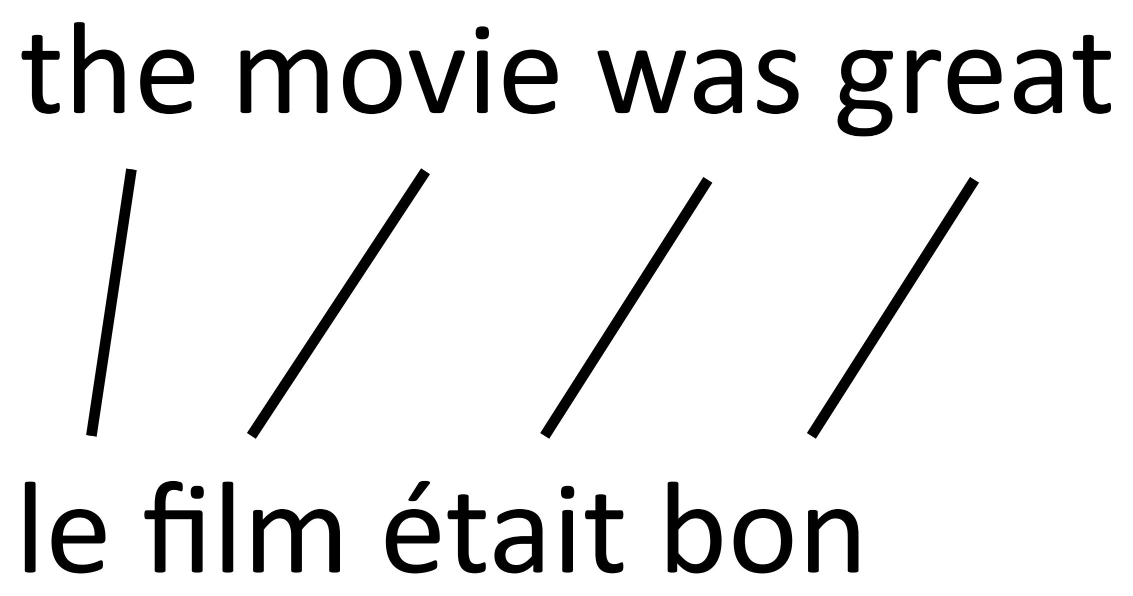
- Much less burden on the hidden state



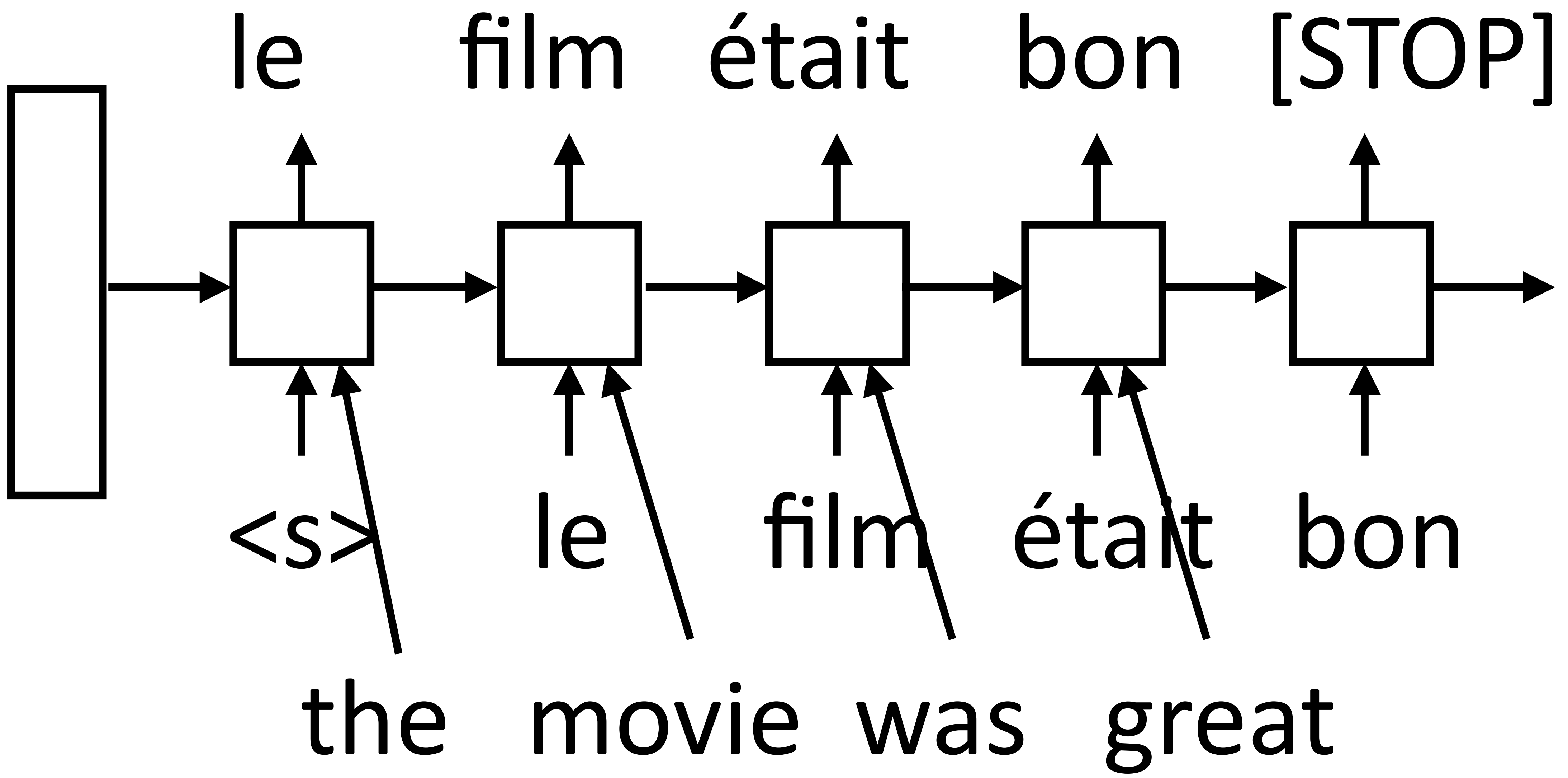


Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated



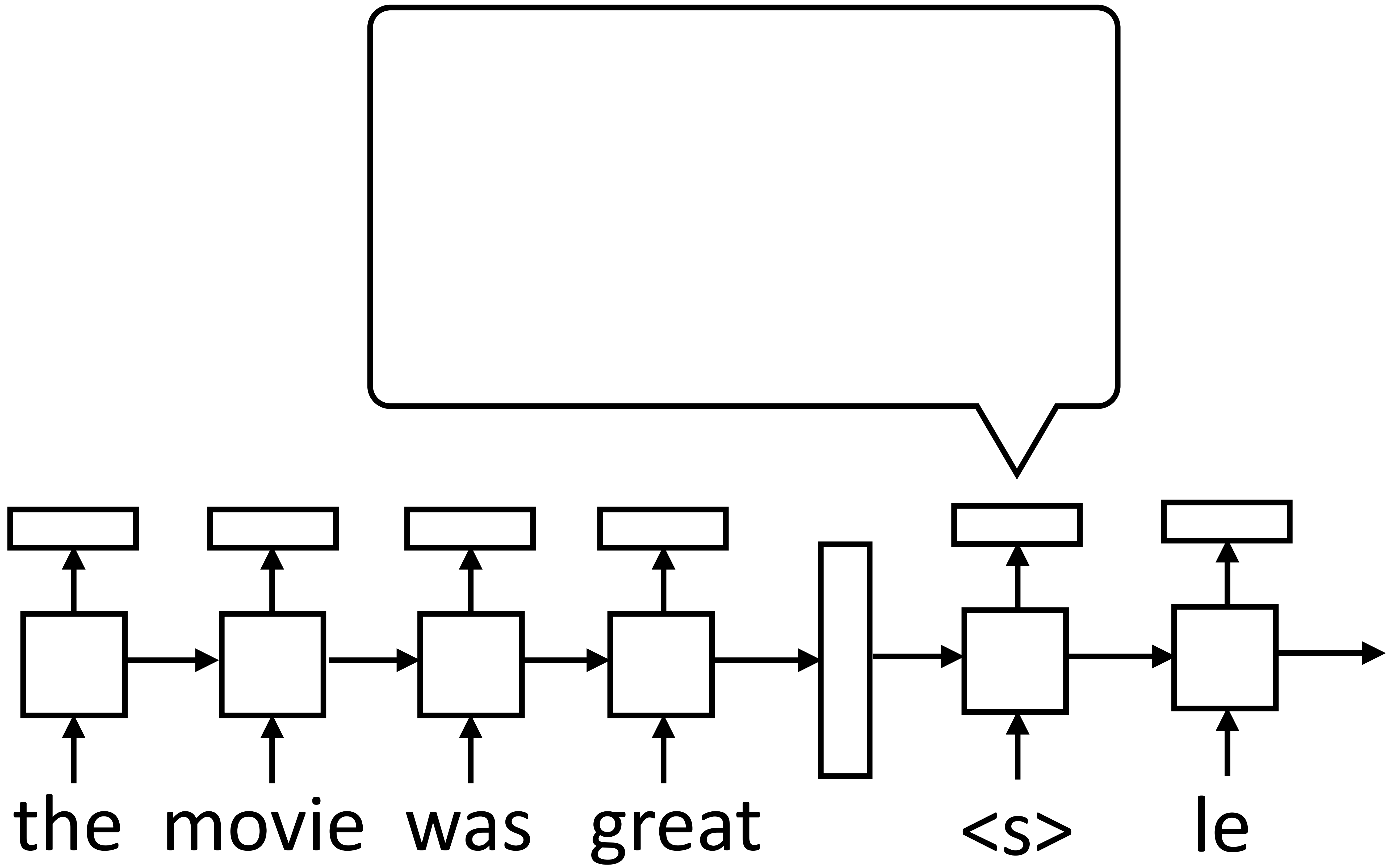
- Can look at the corresponding input word when translating — this could scale!



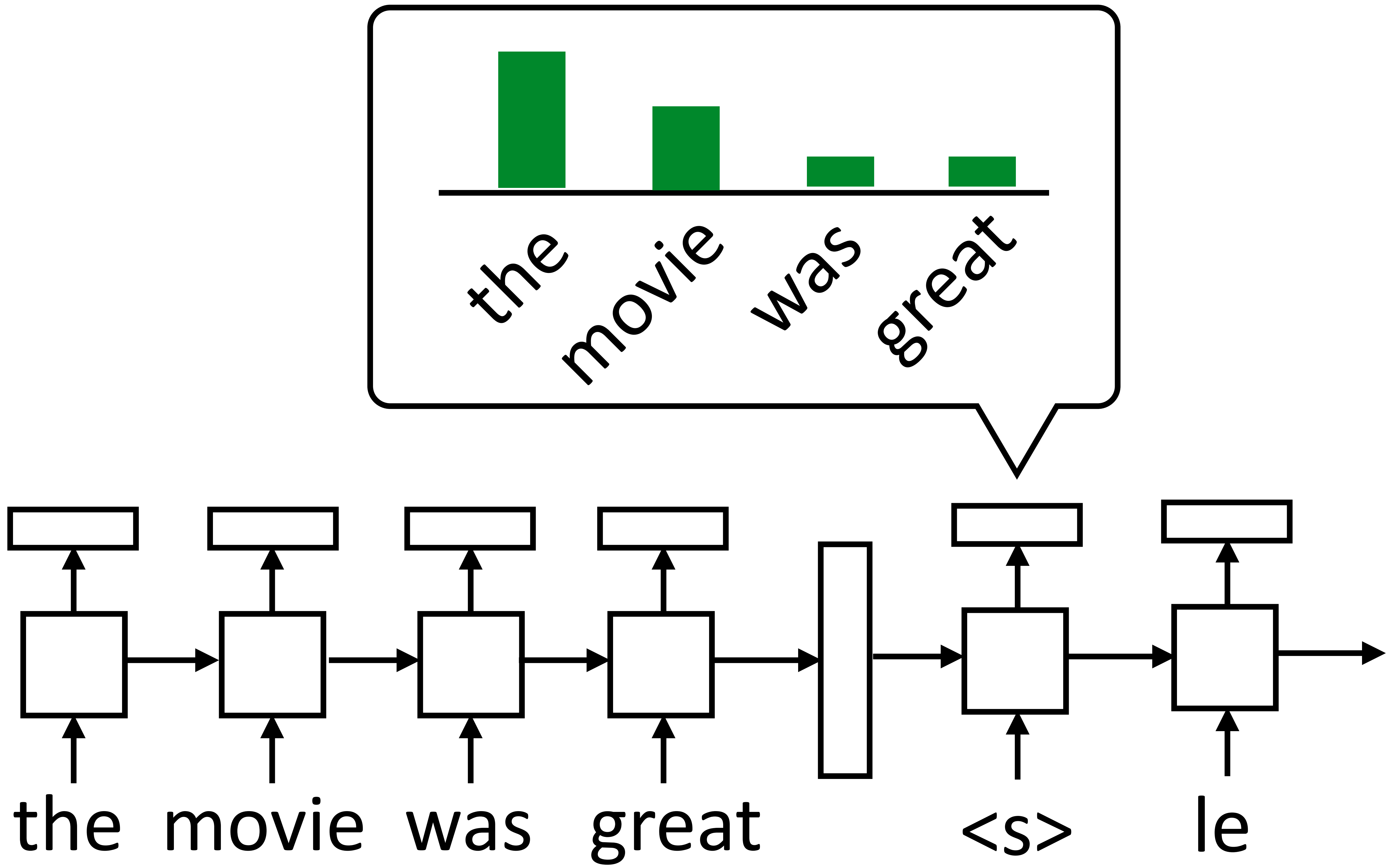
- Much less burden on the hidden state

- How can we achieve this without hardcoding it?

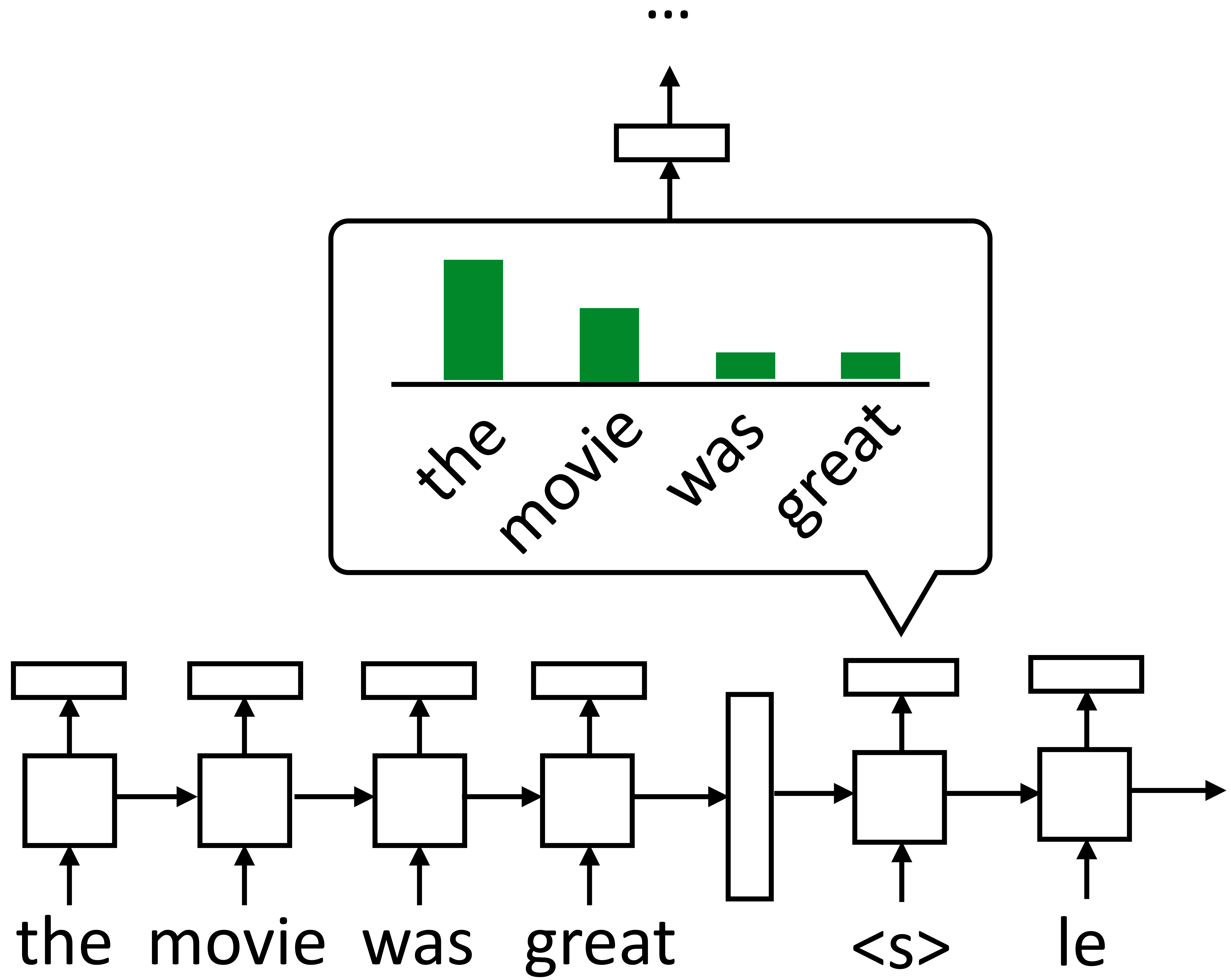
Attention



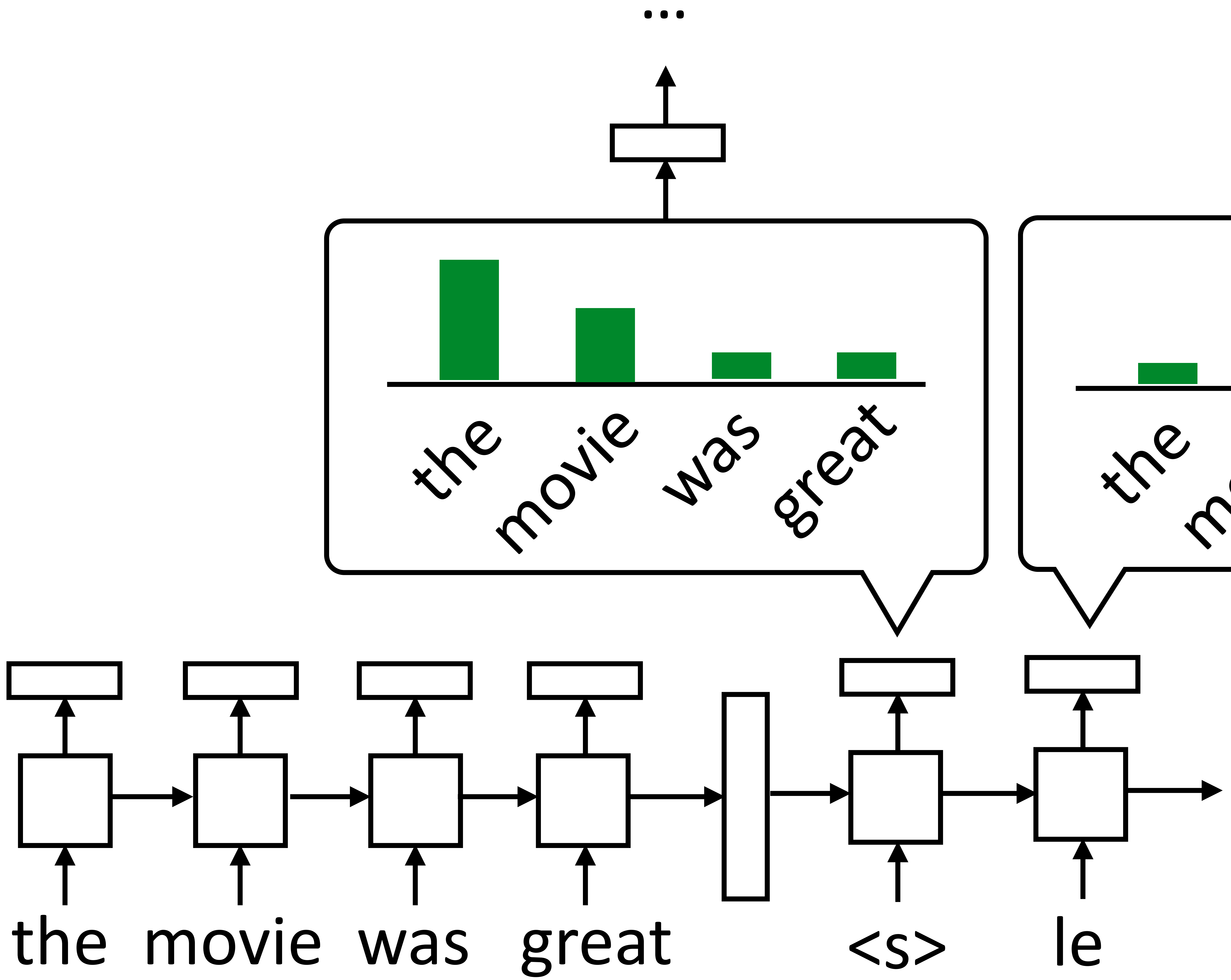
Attention



Attention

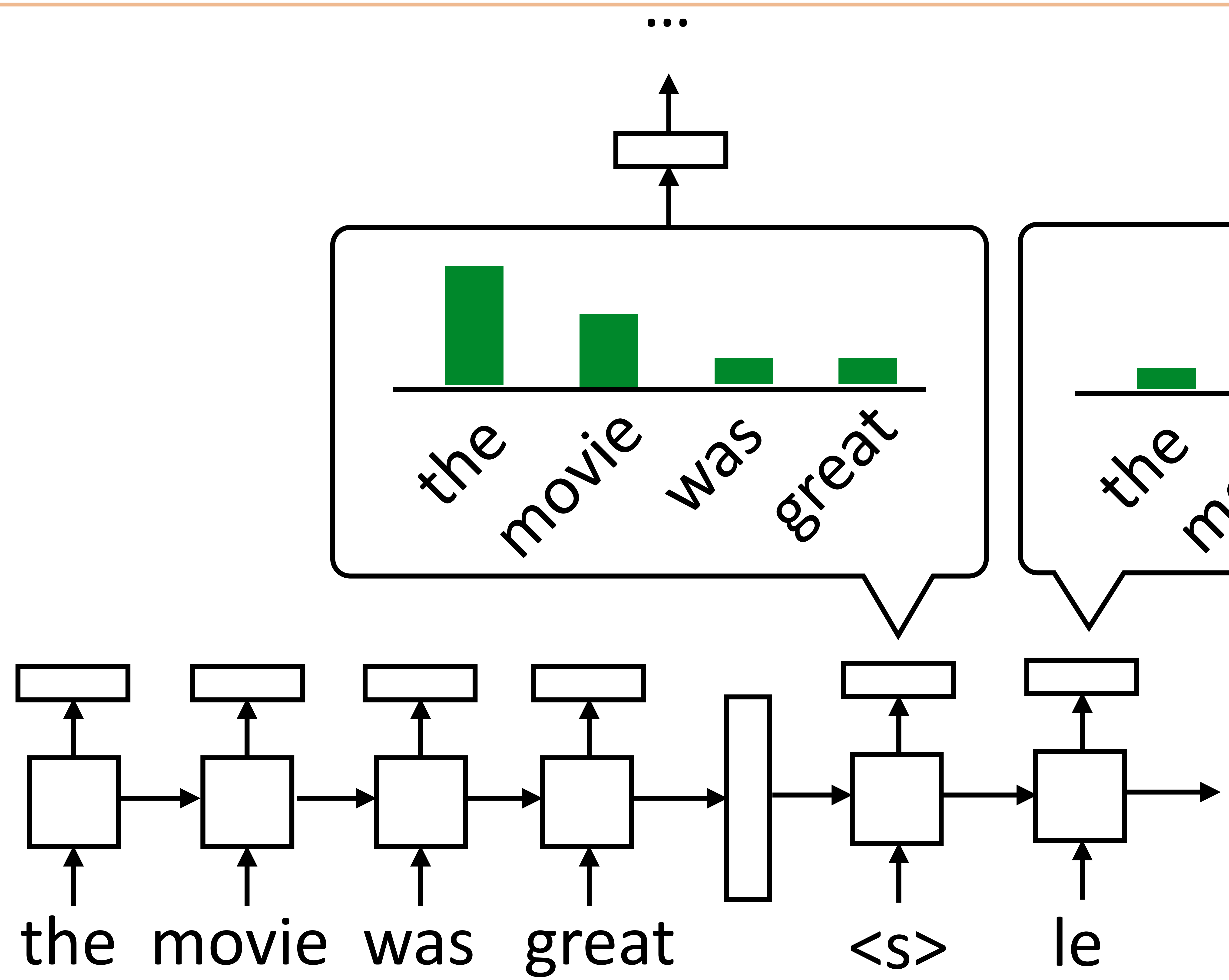


Attention



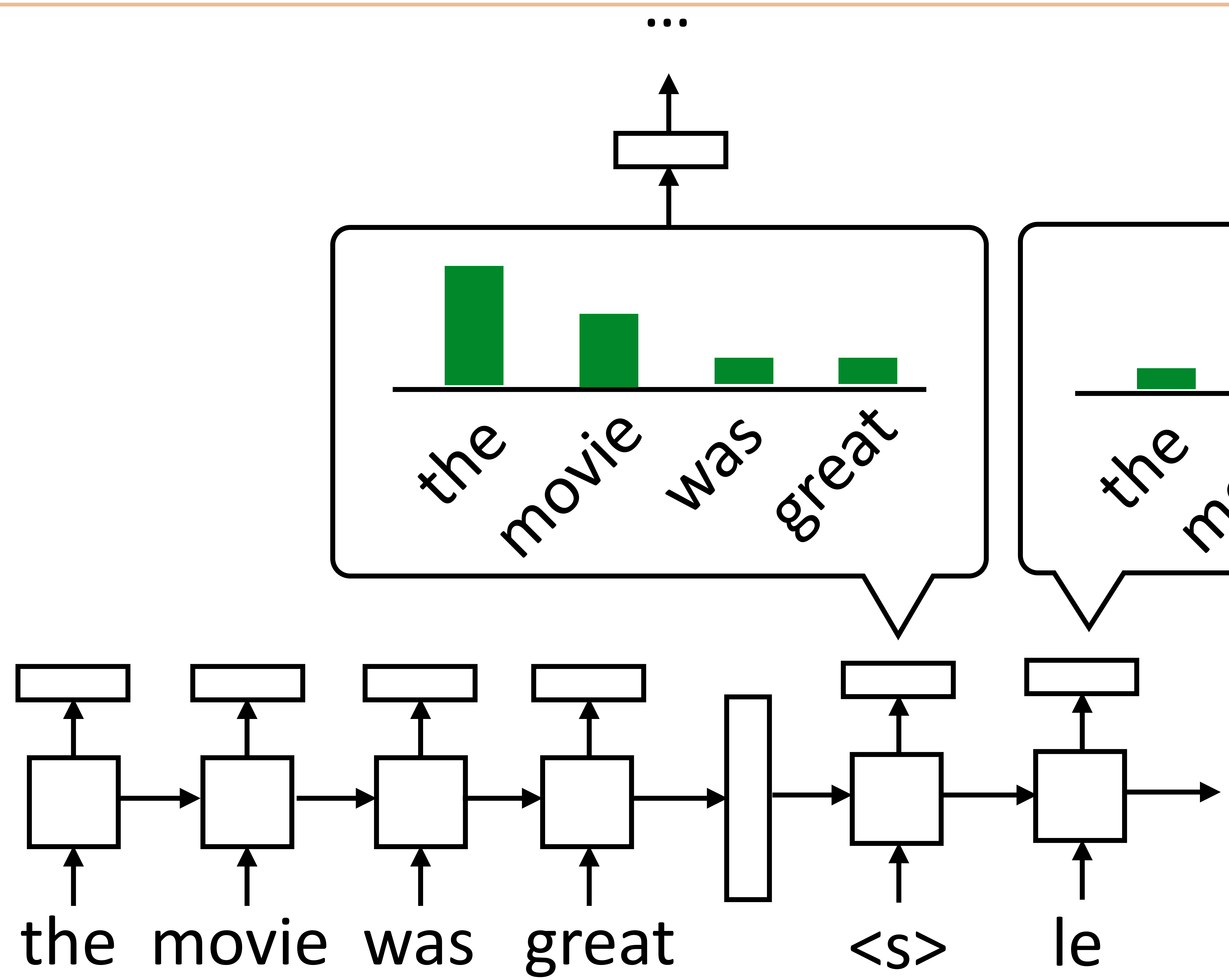
...
the
movie
was
great
the
movie
was
great
the
movie
was
great
<s>
le

Attention



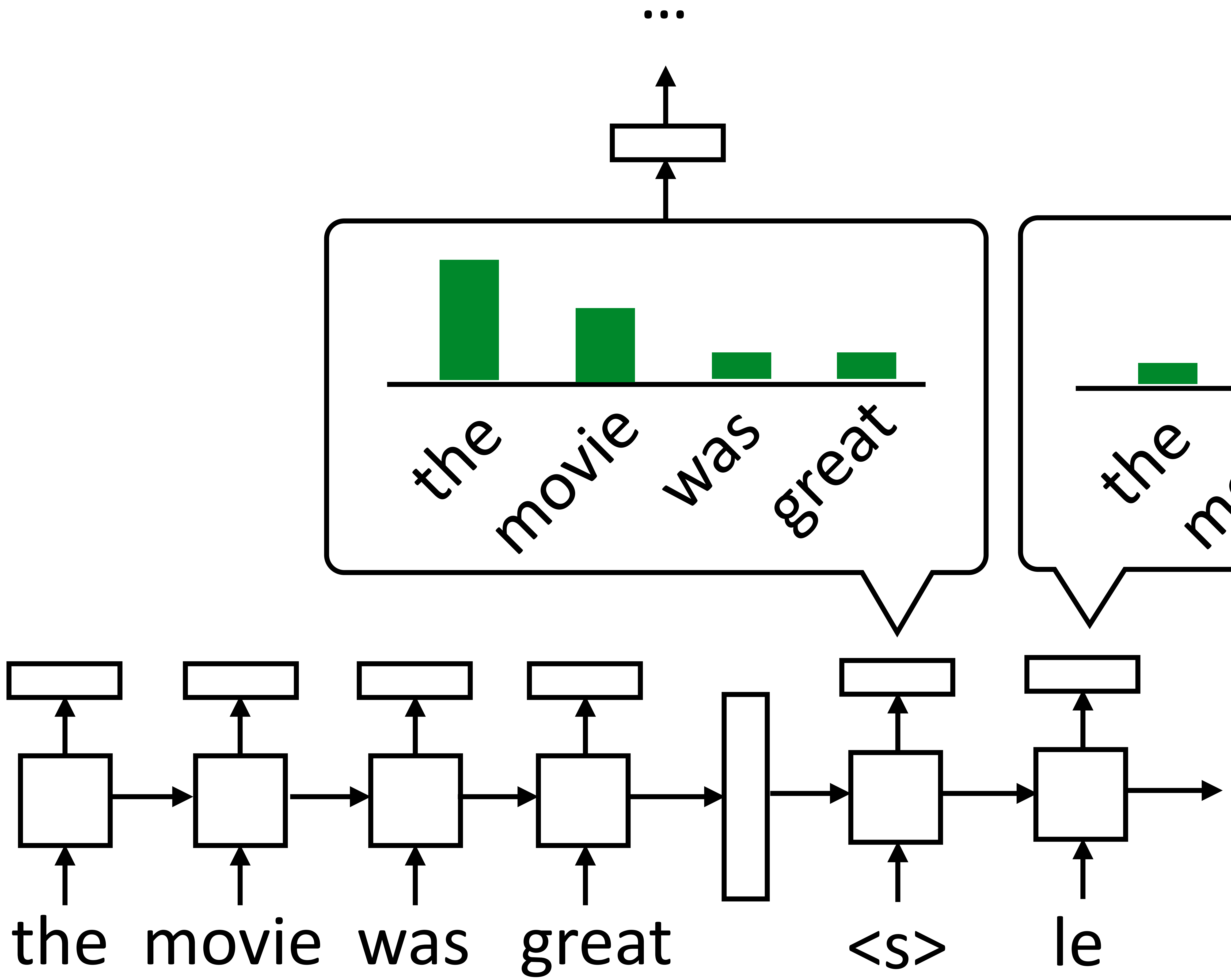
...
...
the
movie
was
great
the
movie
was
great
the
movie
was
great
<s>
le

Attention



- At each decoder state, compute a distribution over source inputs based on current decoder state

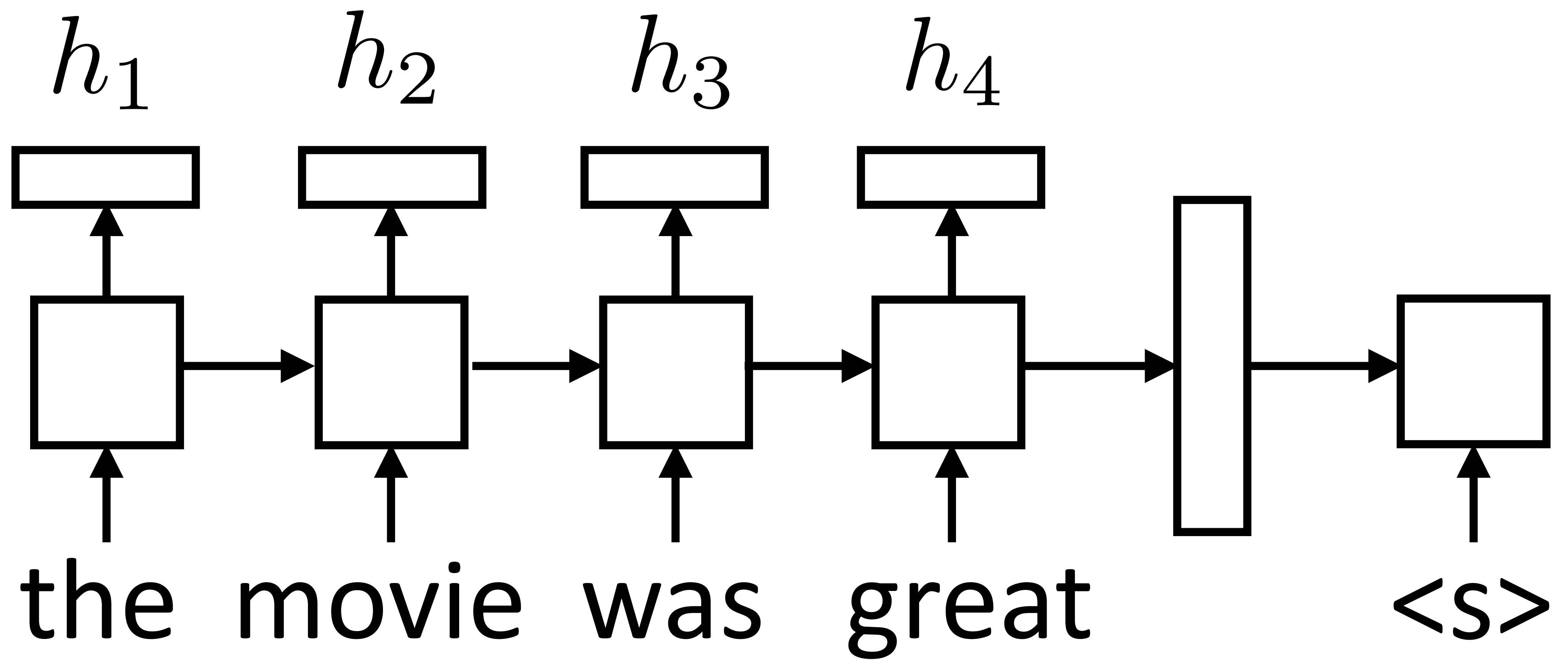
Attention



- At each decoder state, compute a distribution over source inputs based on current decoder state
- Use that in output layer

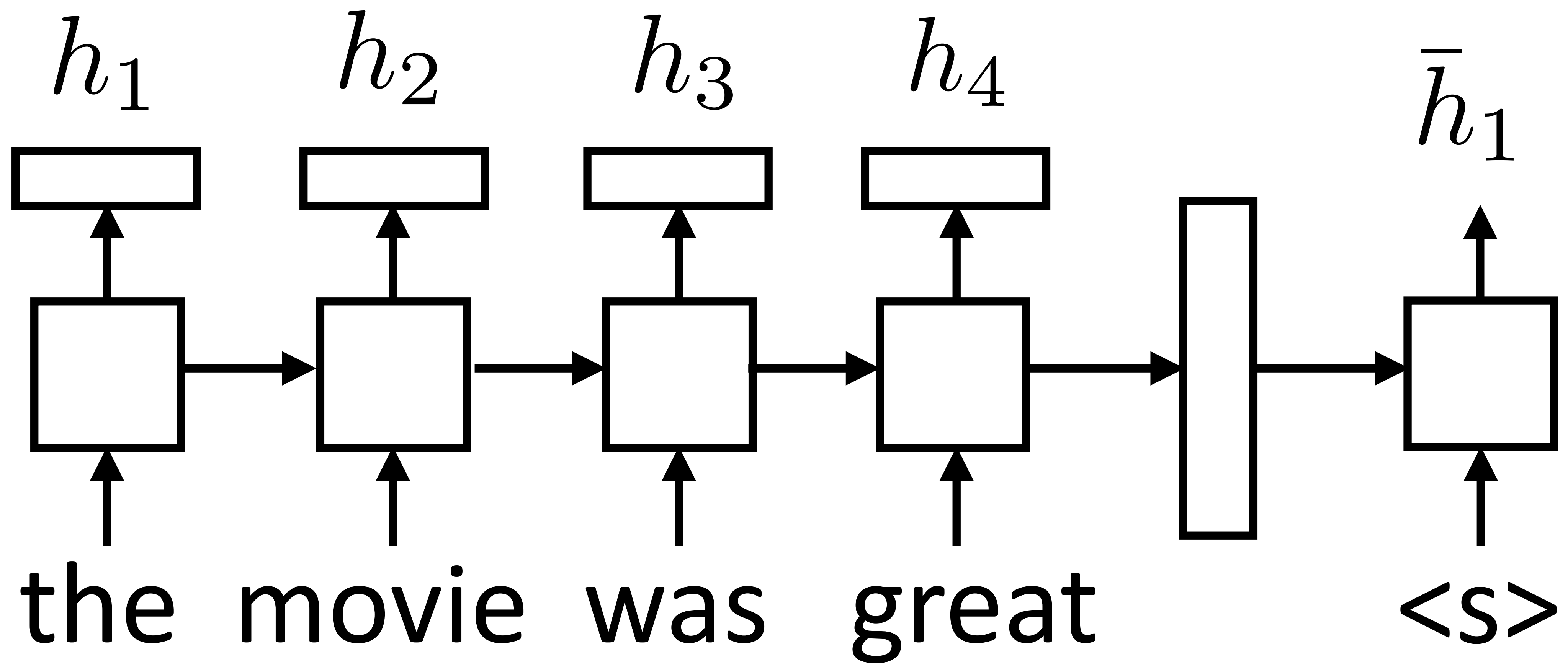
Attention

- For each decoder state, compute weighted sum of input states



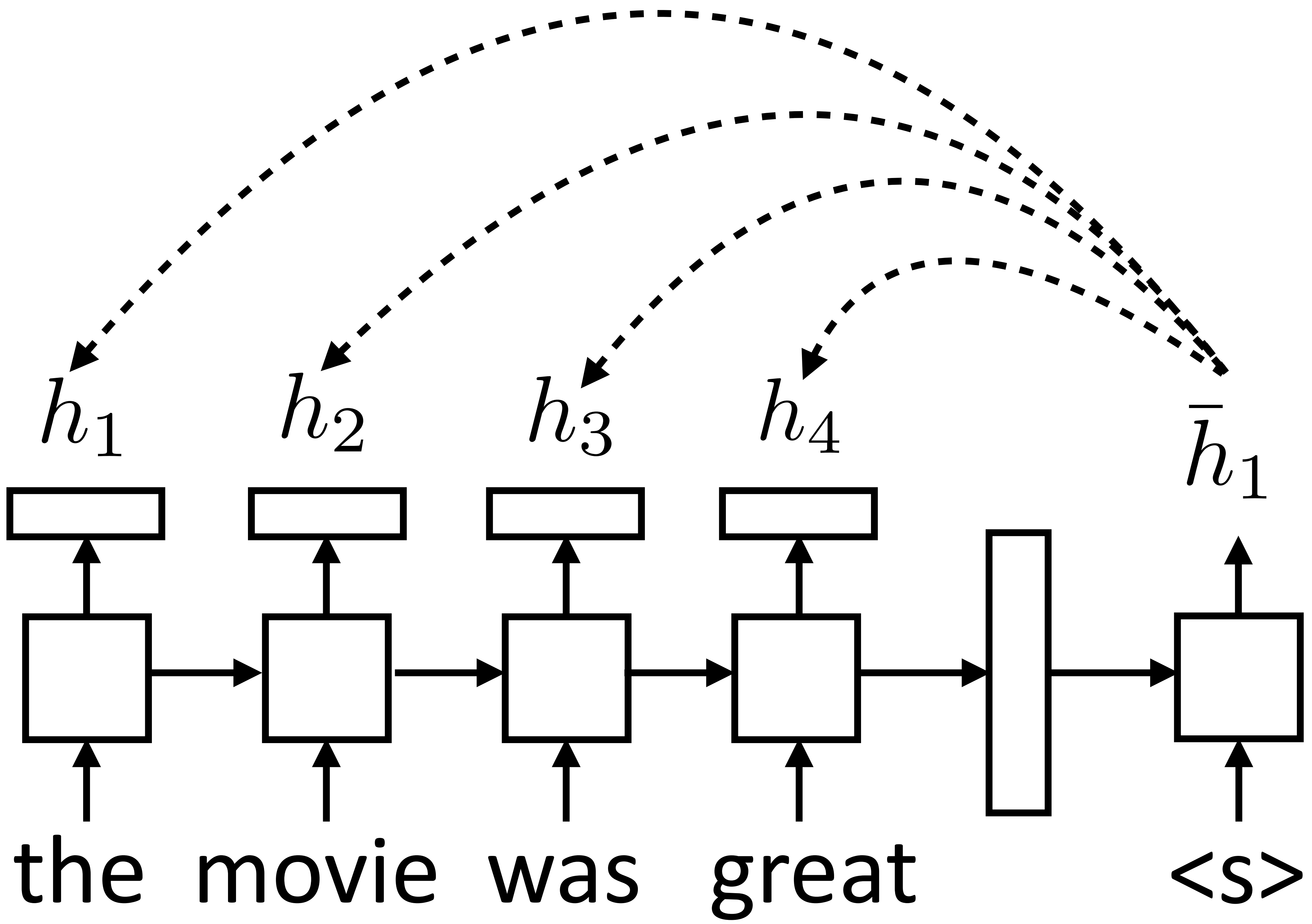
Attention

- For each decoder state, compute weighted sum of input states



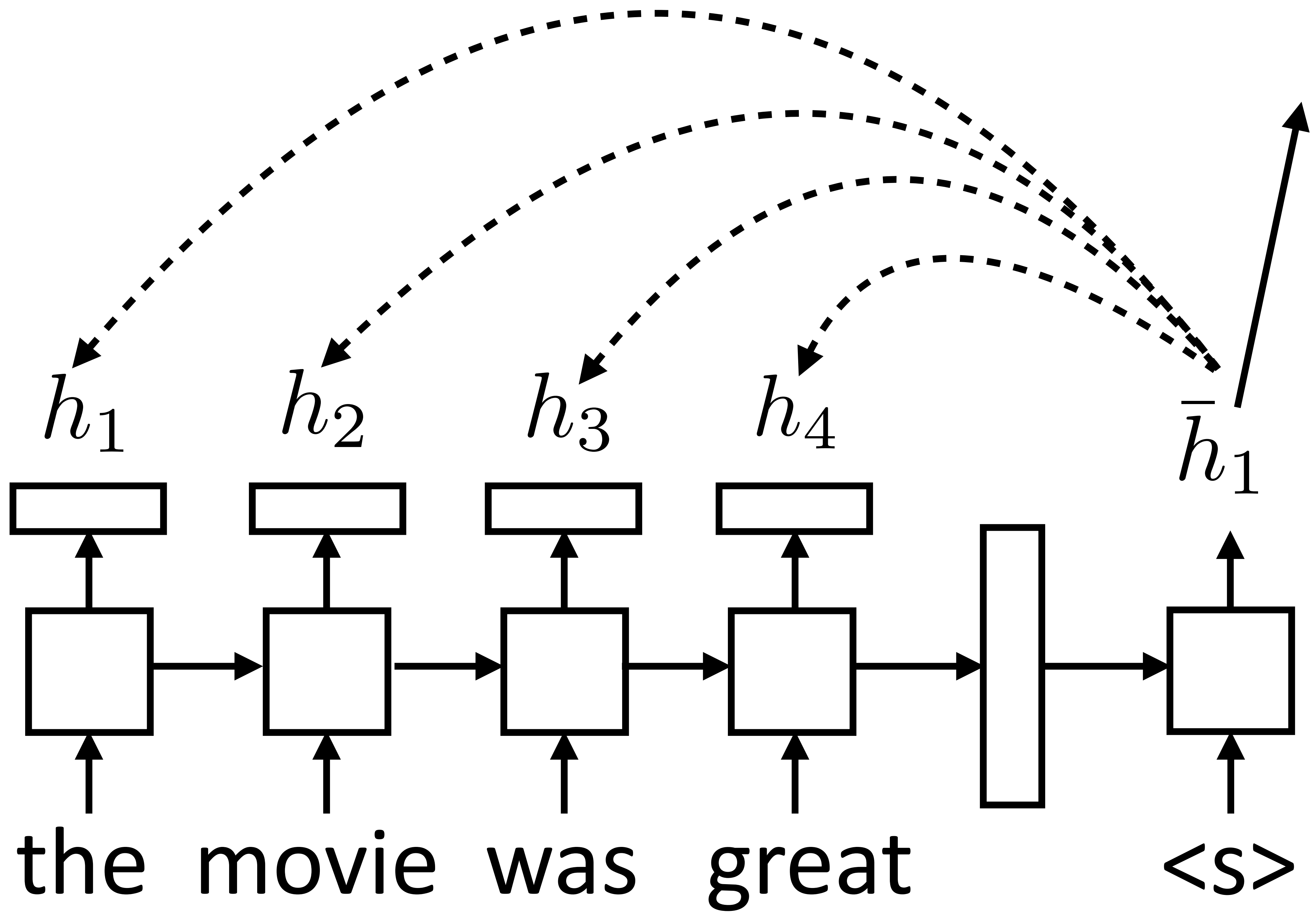
Attention

- For each decoder state, compute weighted sum of input states



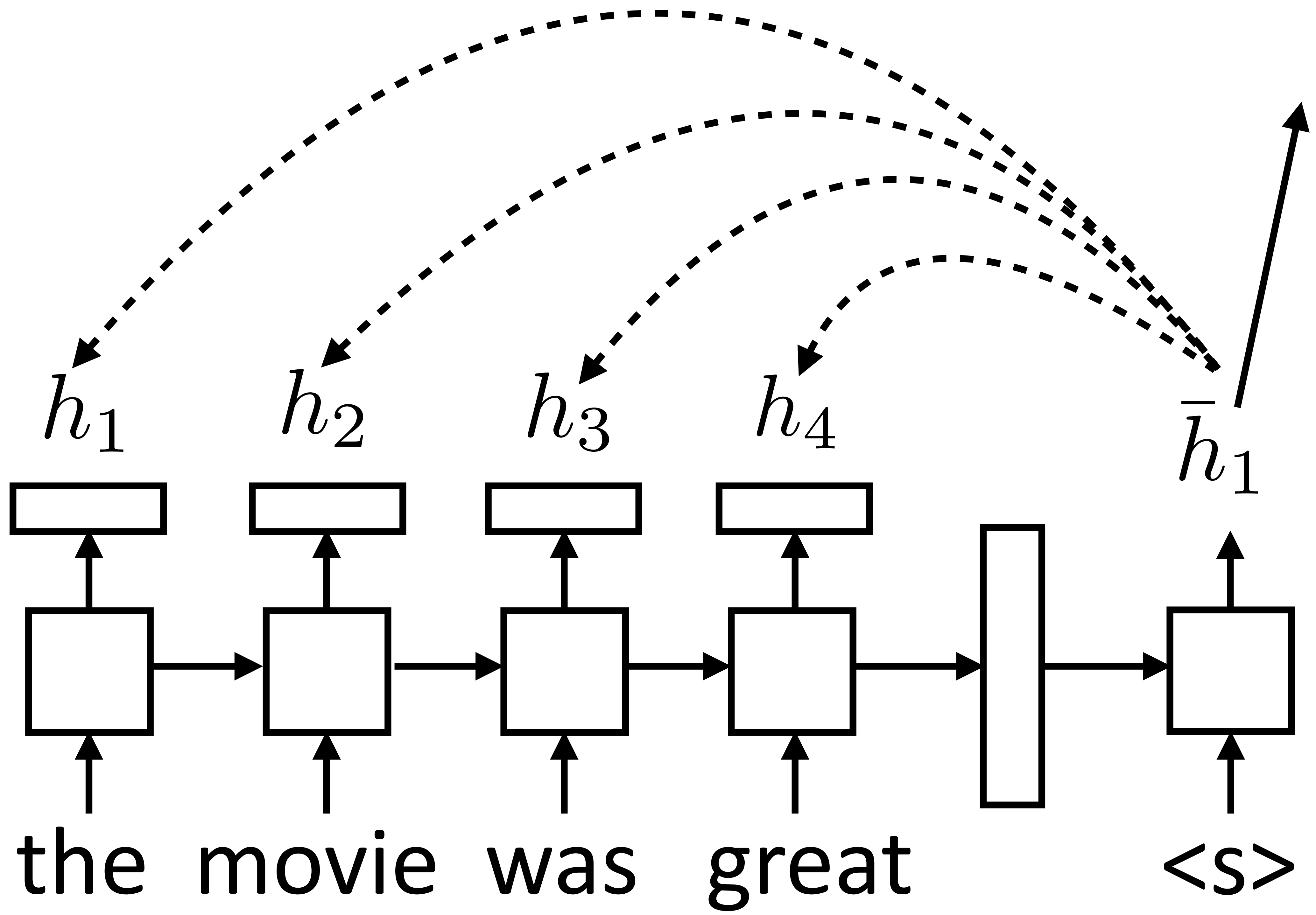
Attention

- For each decoder state, compute weighted sum of input states



Attention

- For each decoder state, compute weighted sum of input states

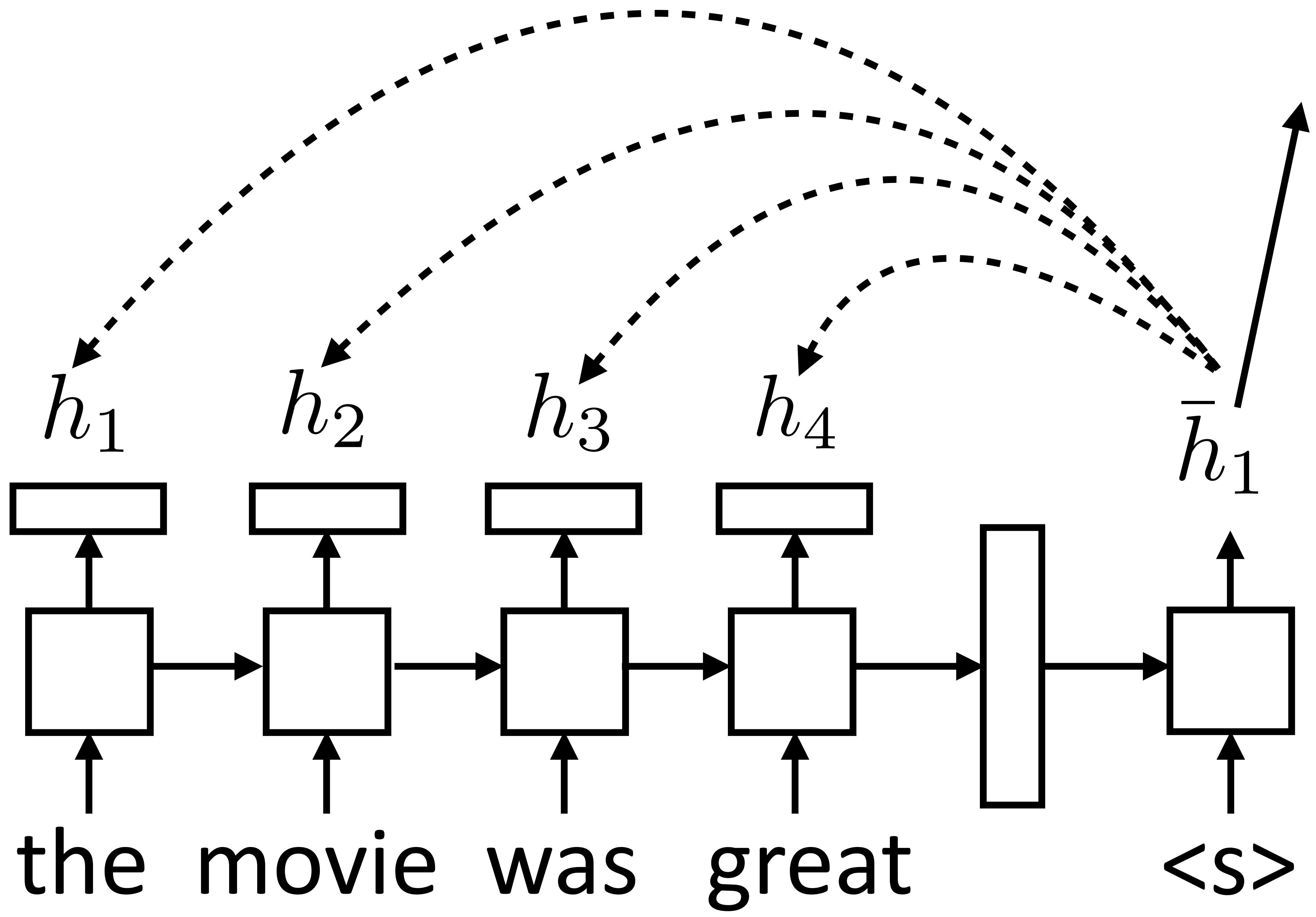


$$e_{ij} = f(\bar{h}_i, h_j)$$

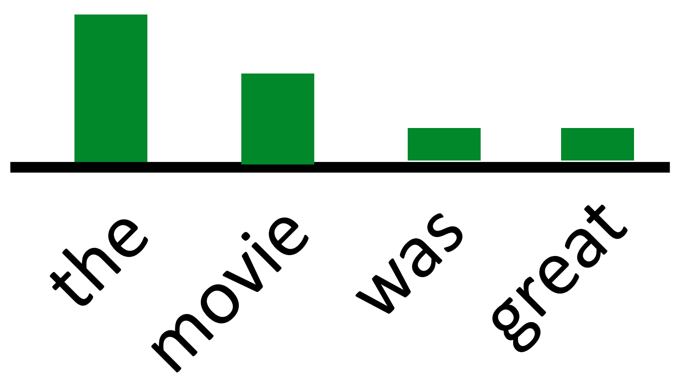
- Unnormalized scalar weight

Attention

- For each decoder state, compute weighted sum of input states



$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

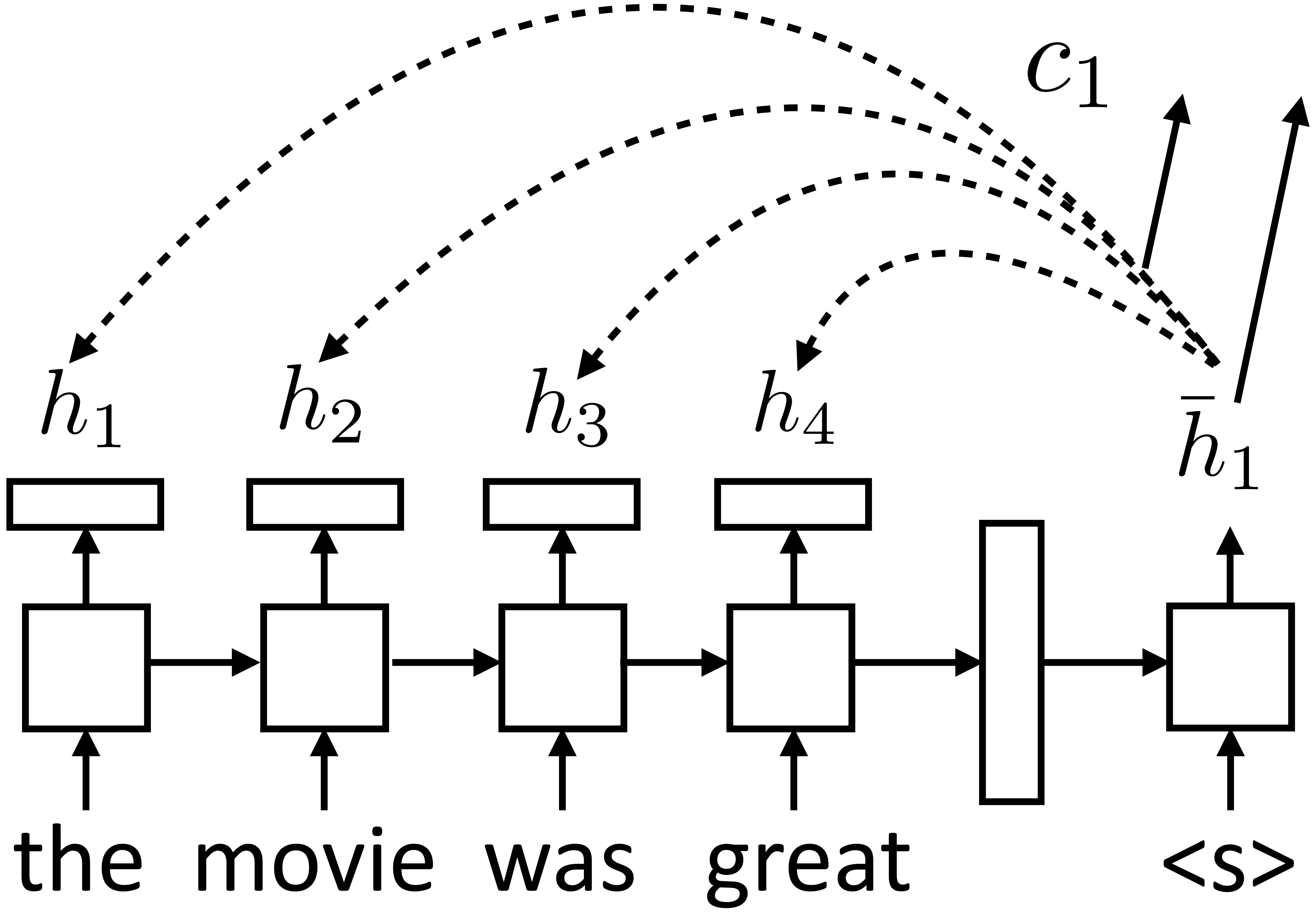


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

Attention

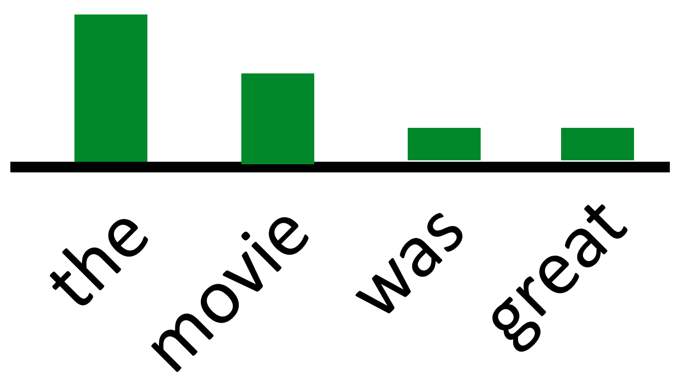
- For each decoder state, compute weighted sum of input states



$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

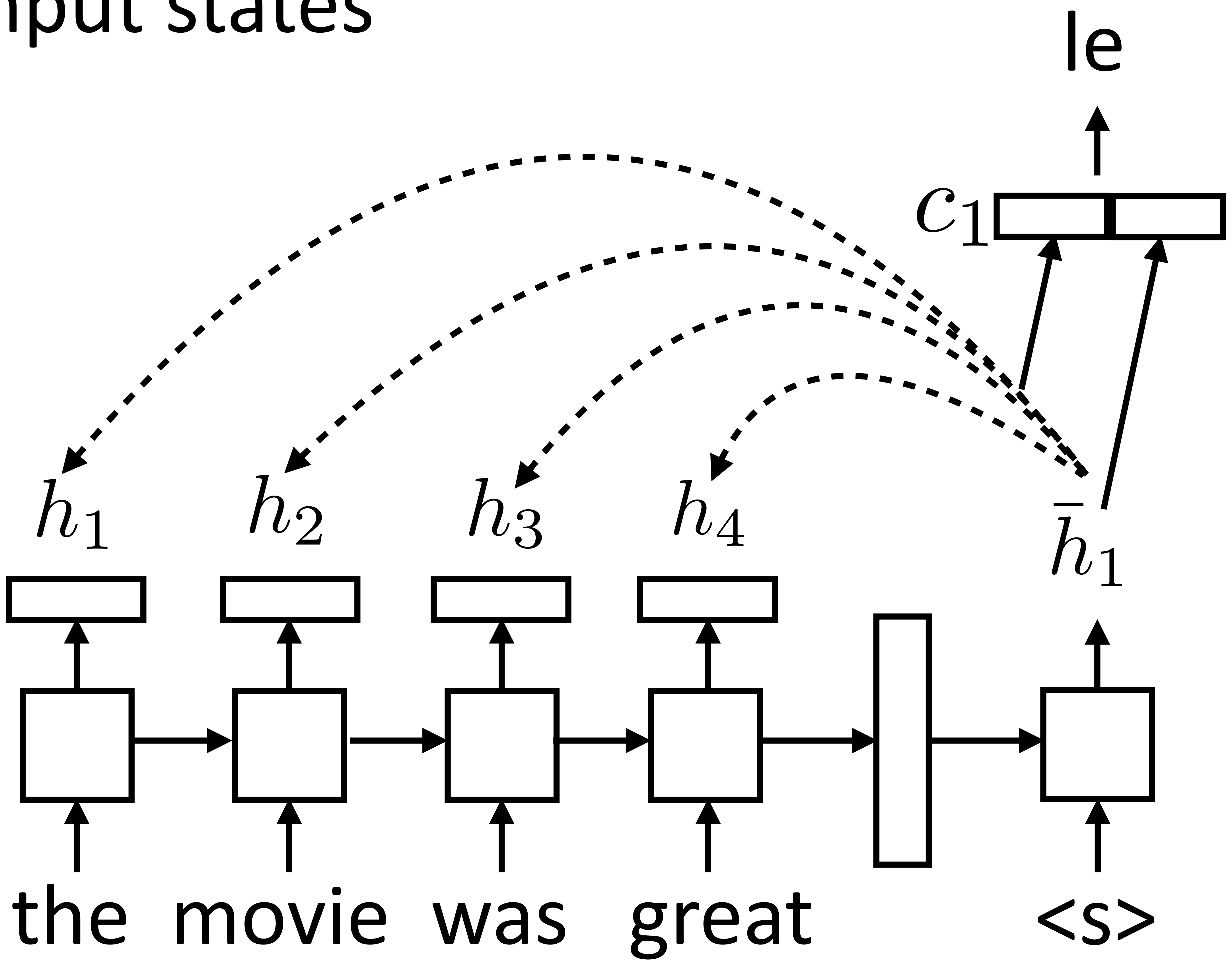


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

Attention

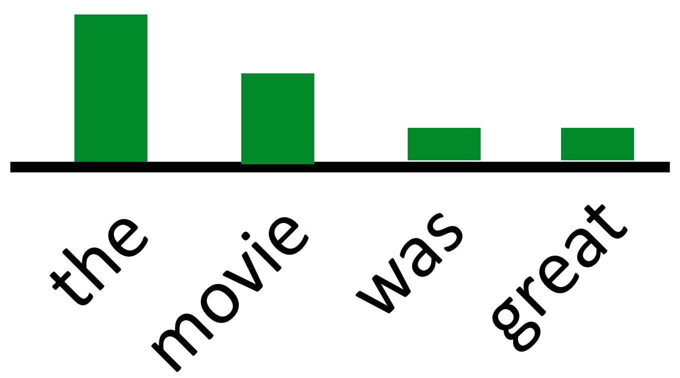
- For each decoder state, compute weighted sum of input states



$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

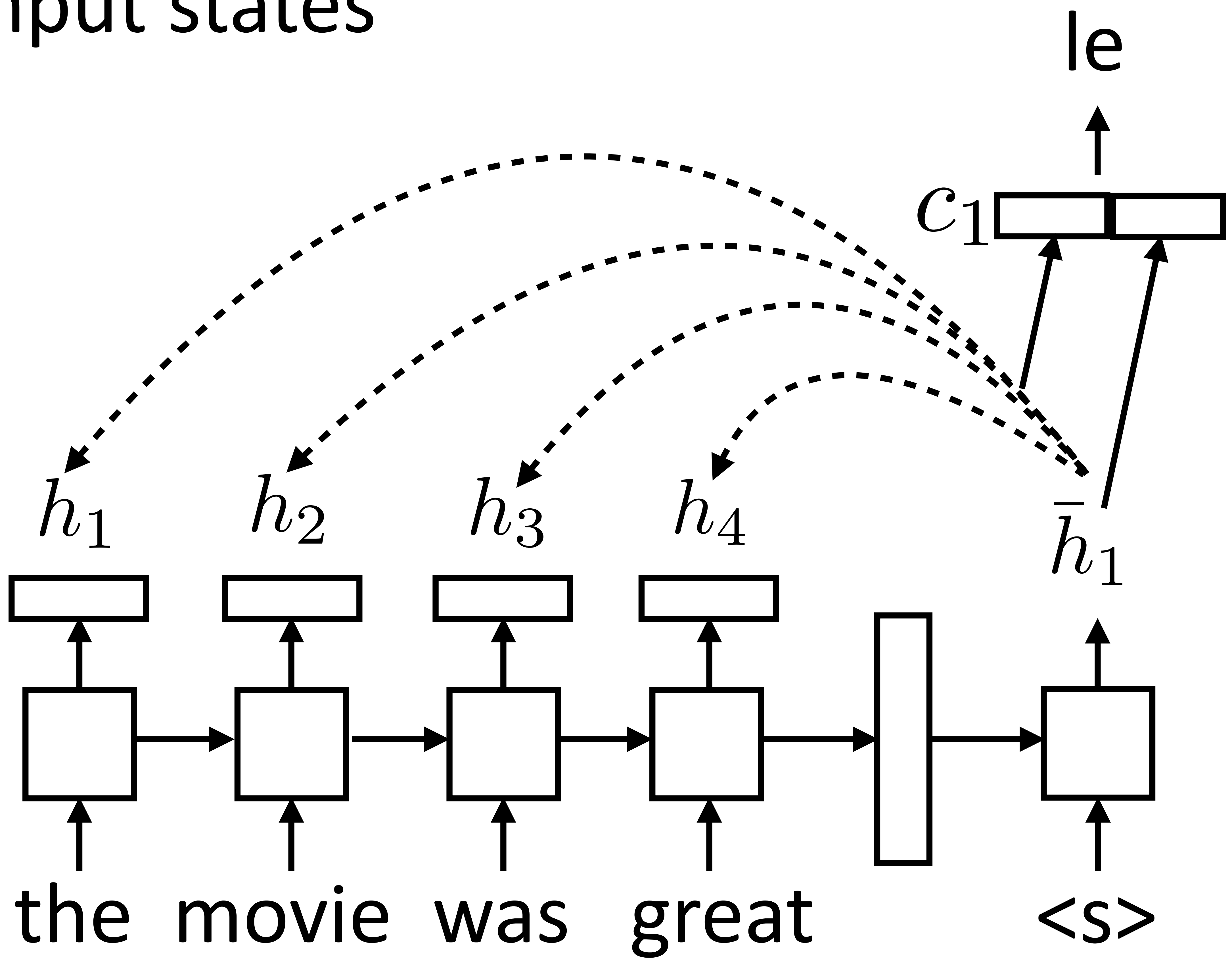


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

Attention

- For each decoder state, compute weighted sum of input states

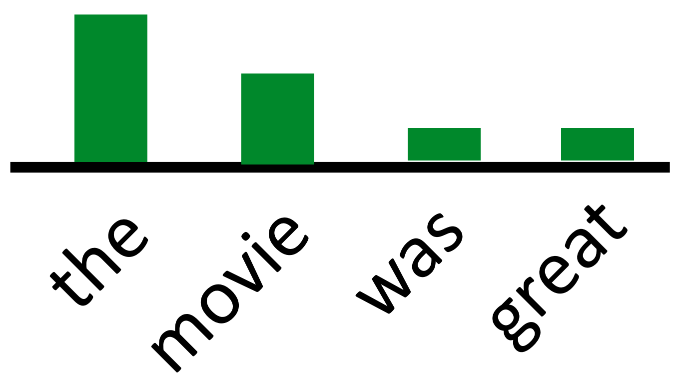


$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i])$$

$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

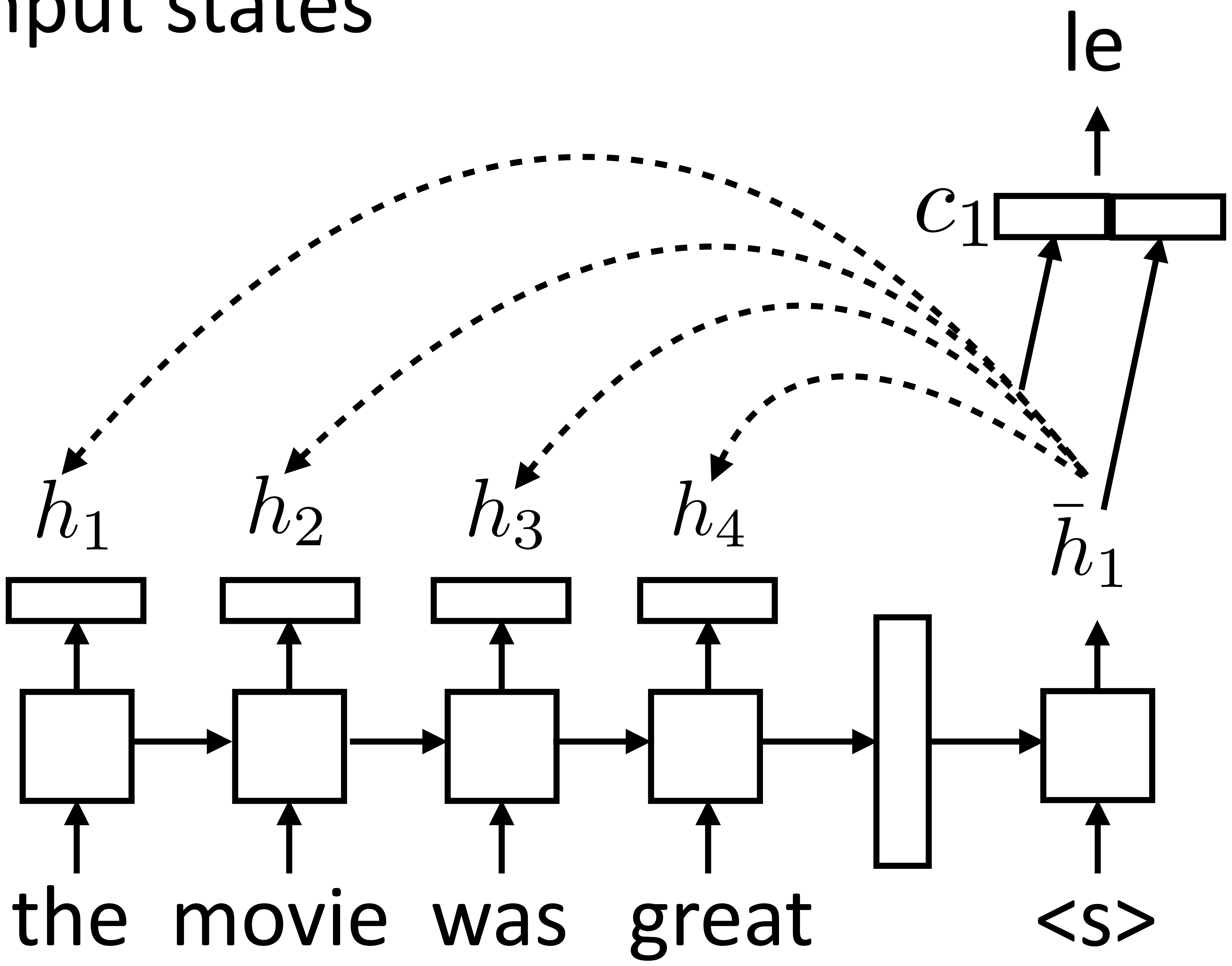


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

Attention

- For each decoder state, compute weighted sum of input states



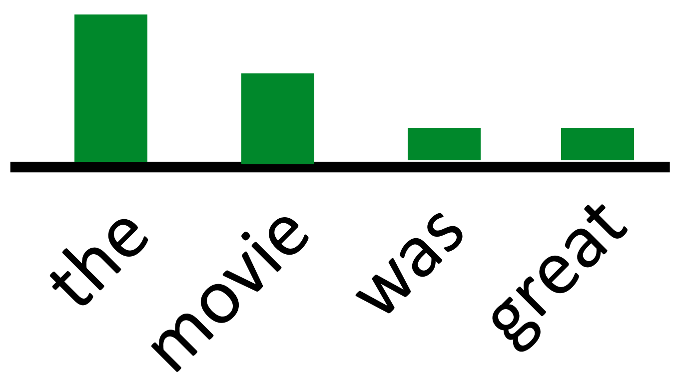
- No attn: $P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h}_i)$

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W[c_i; \bar{h}_i])$$

$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

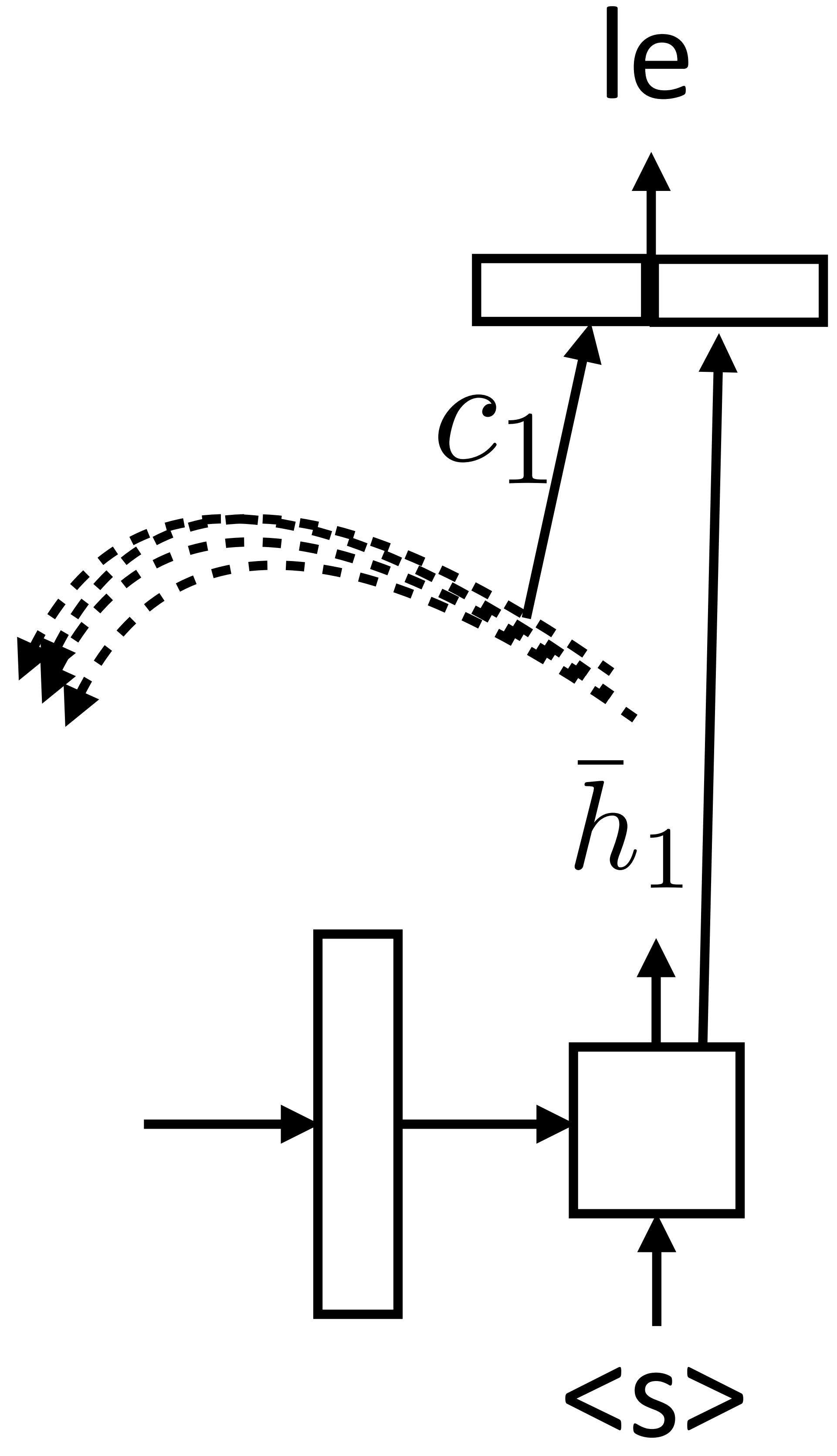
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$



$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

Attention



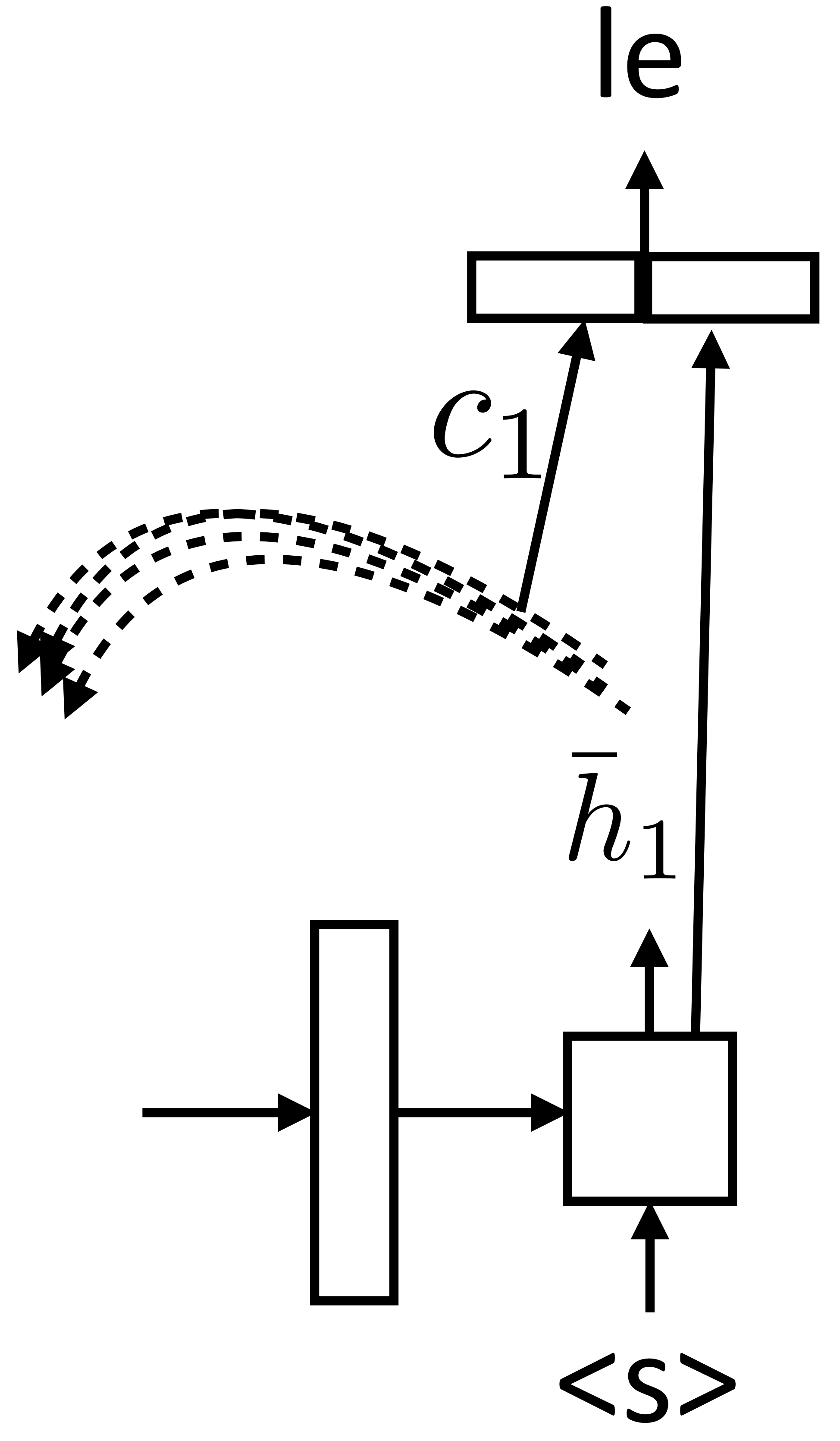
$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

Luong et al. (2015)

Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

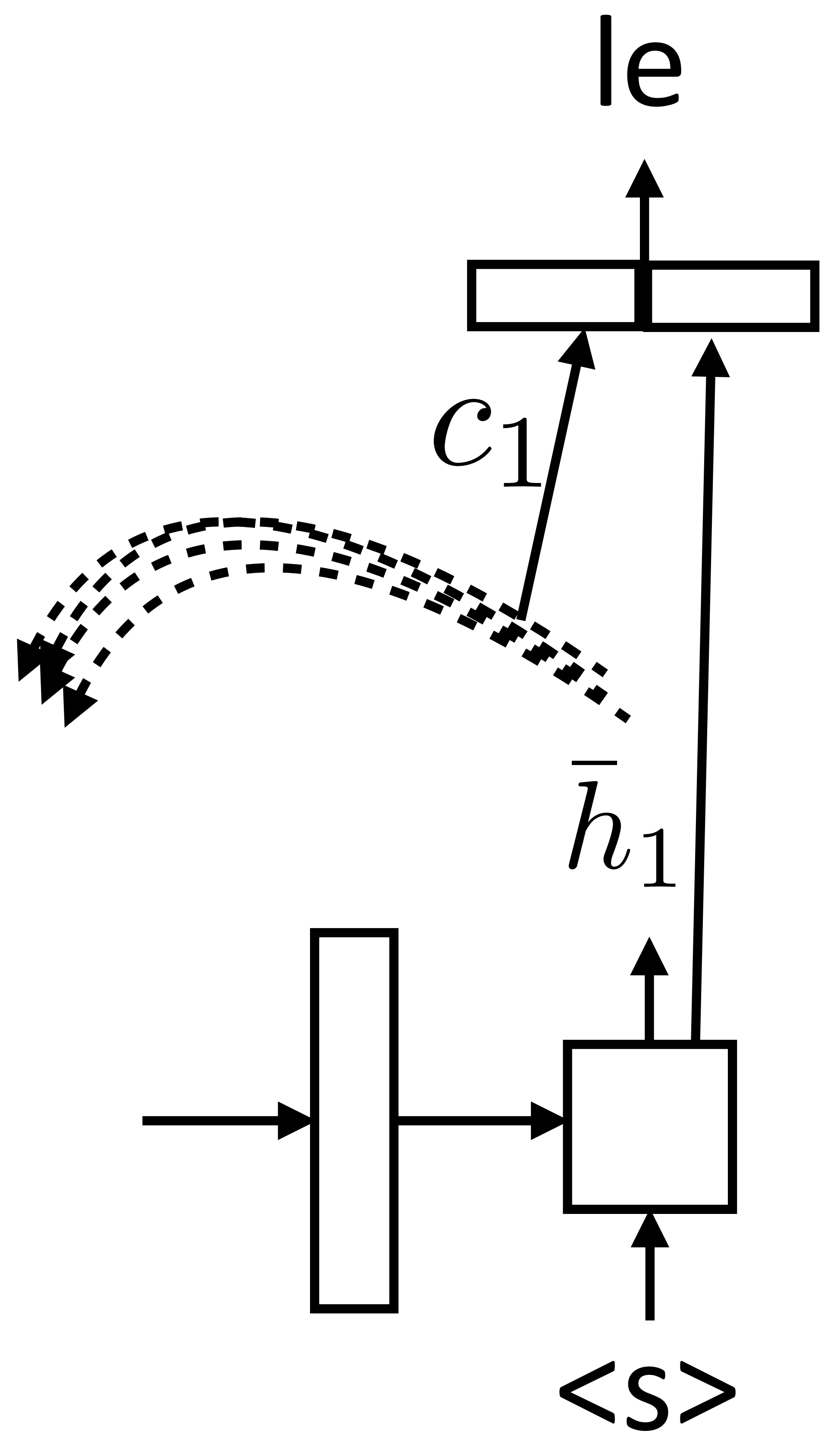
$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

- Bahdanau+ (2014): additive

Luong et al. (2015)

Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

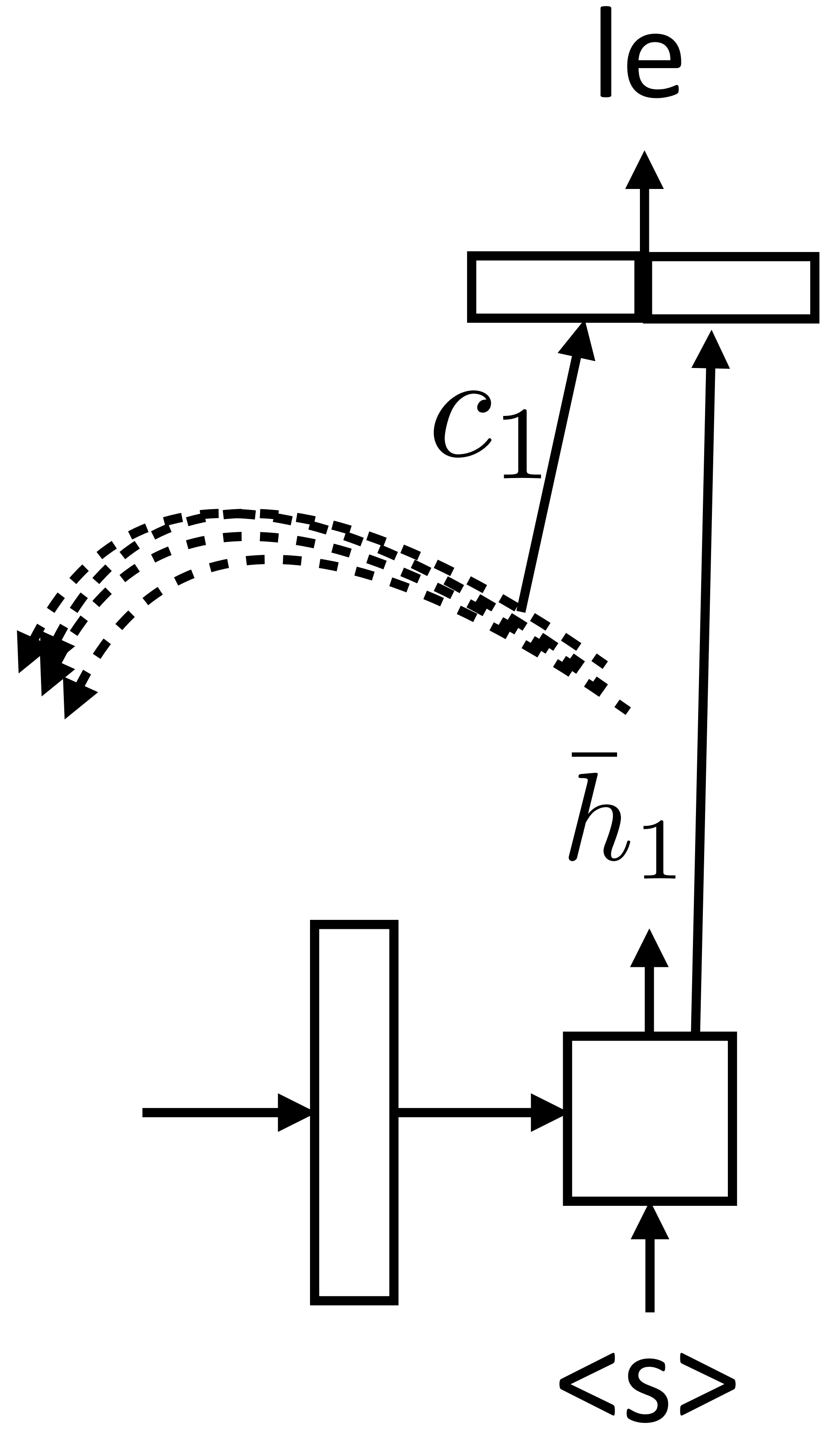
- Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

- Luong+ (2015): dot product

Luong et al. (2015)

Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

- Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

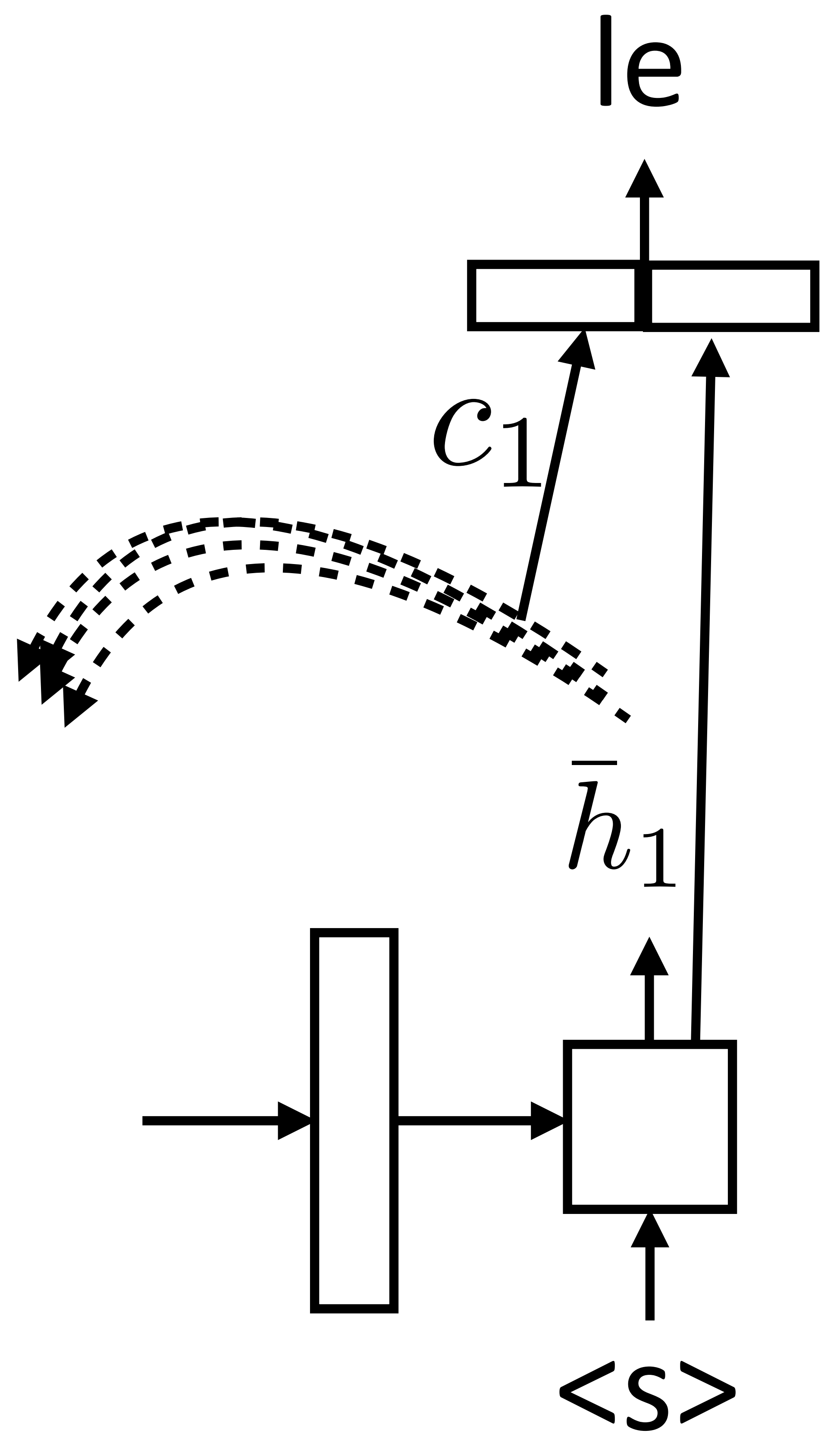
- Luong+ (2015): dot product

$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

- Luong+ (2015): bilinear

Luong et al. (2015)

Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

- Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

- Luong+ (2015): dot product

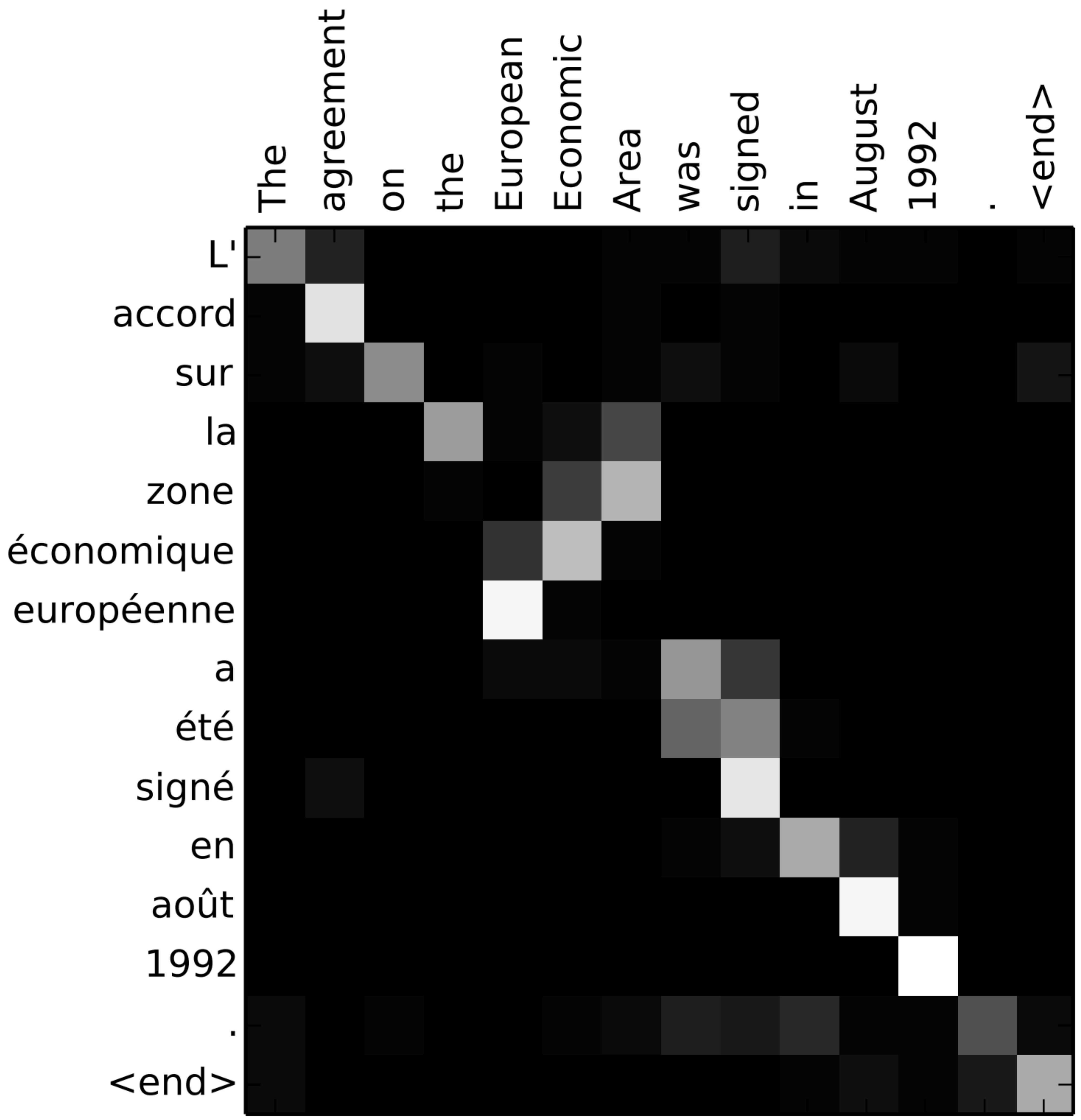
$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

- Luong+ (2015): bilinear

- Note that this all uses outputs of hidden layers

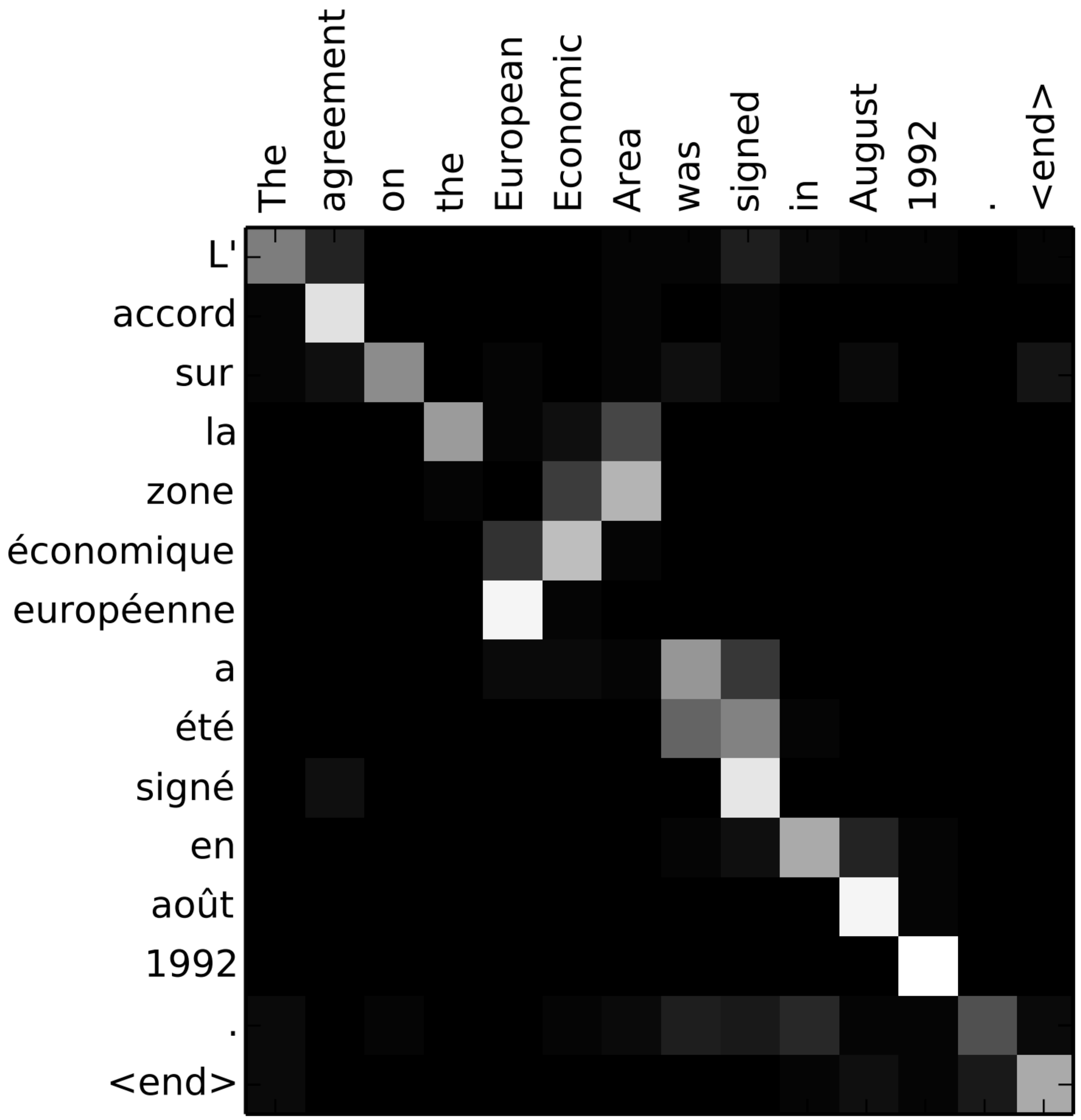
Luong et al. (2015)

Attention


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>

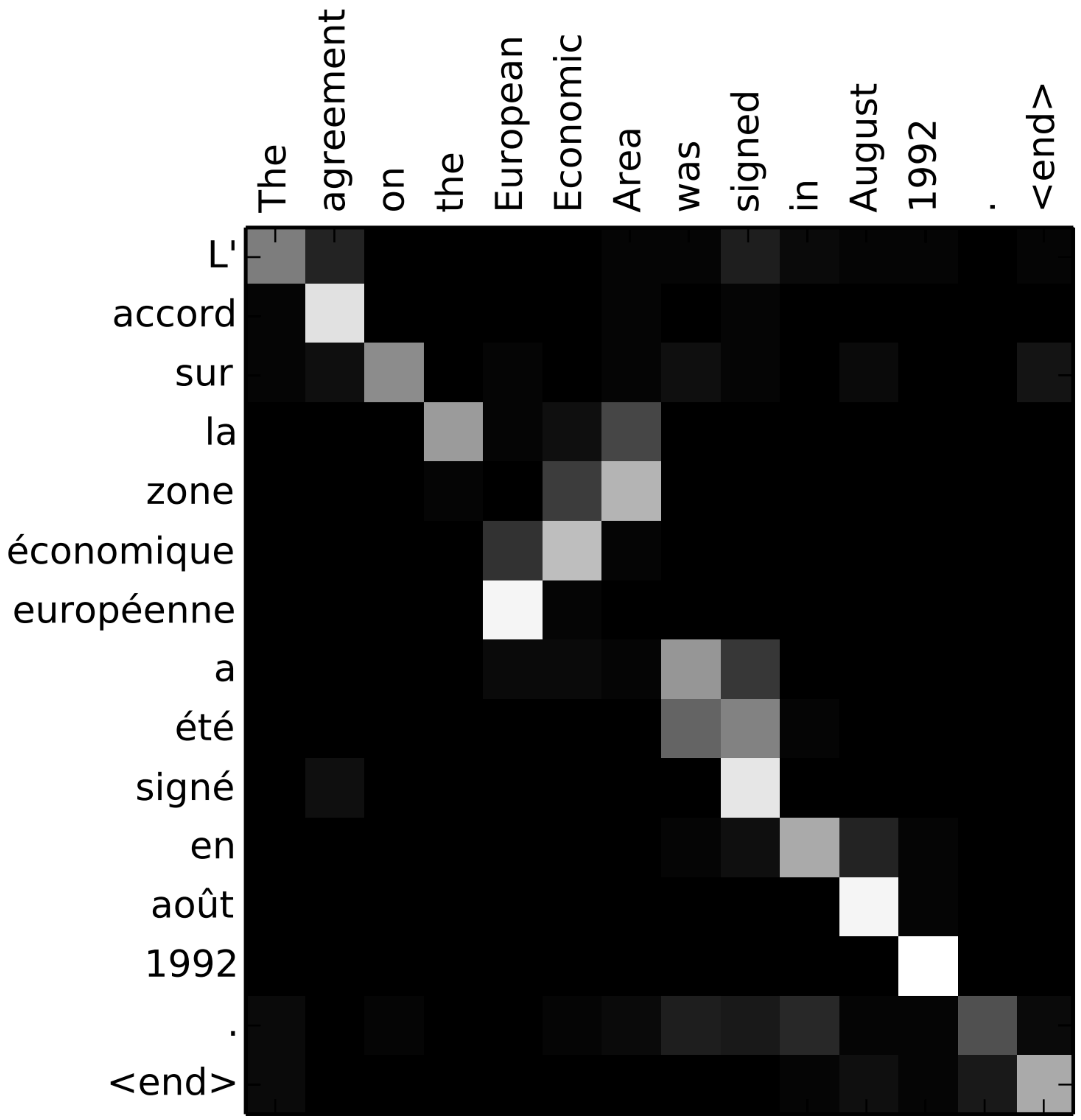
Attention

- Encoder hidden states capture contextual source word identity



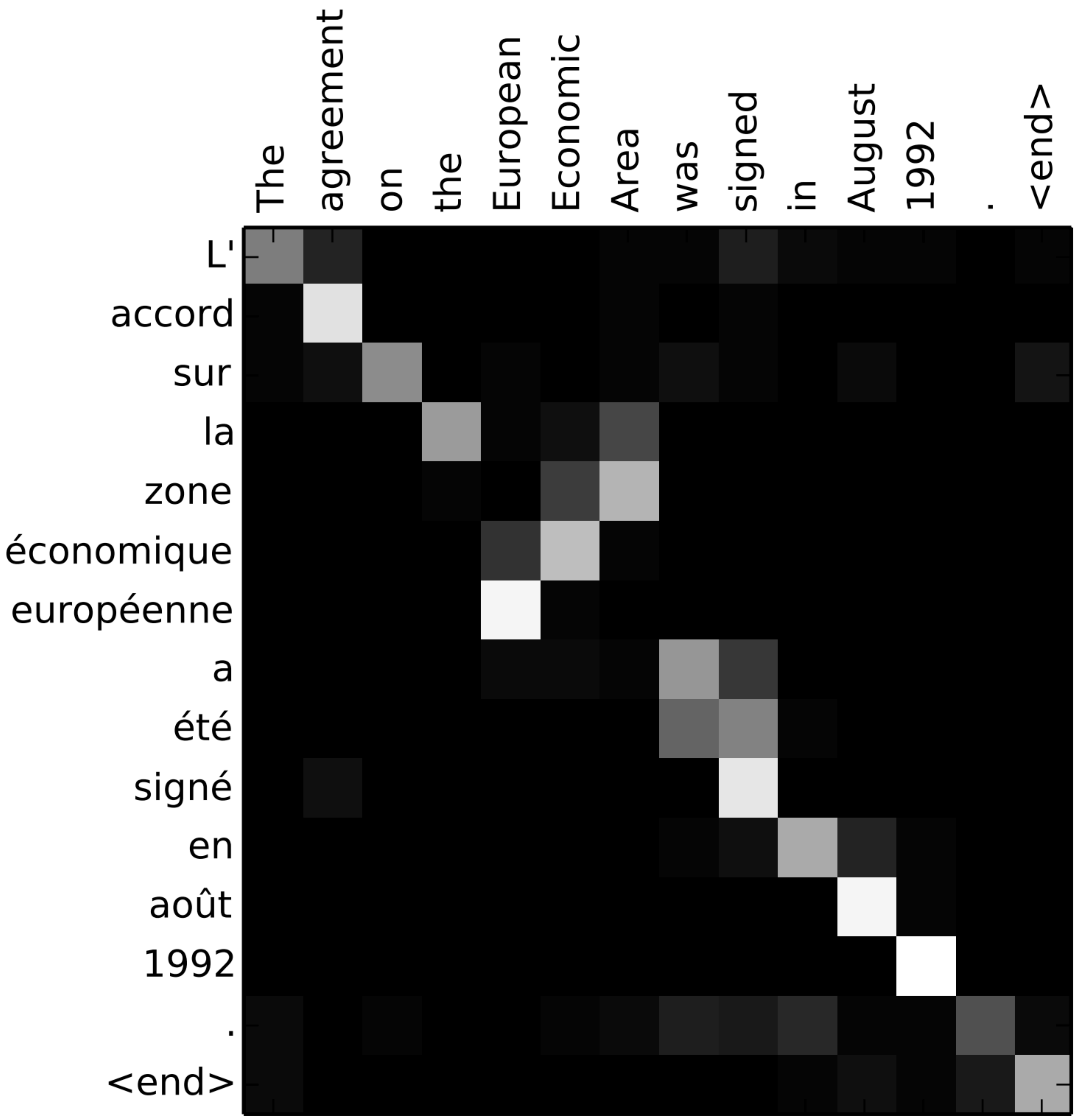
Attention

- Encoder hidden states capture contextual source word identity
- Decoder hidden states are now mostly responsible for selecting what to attend to

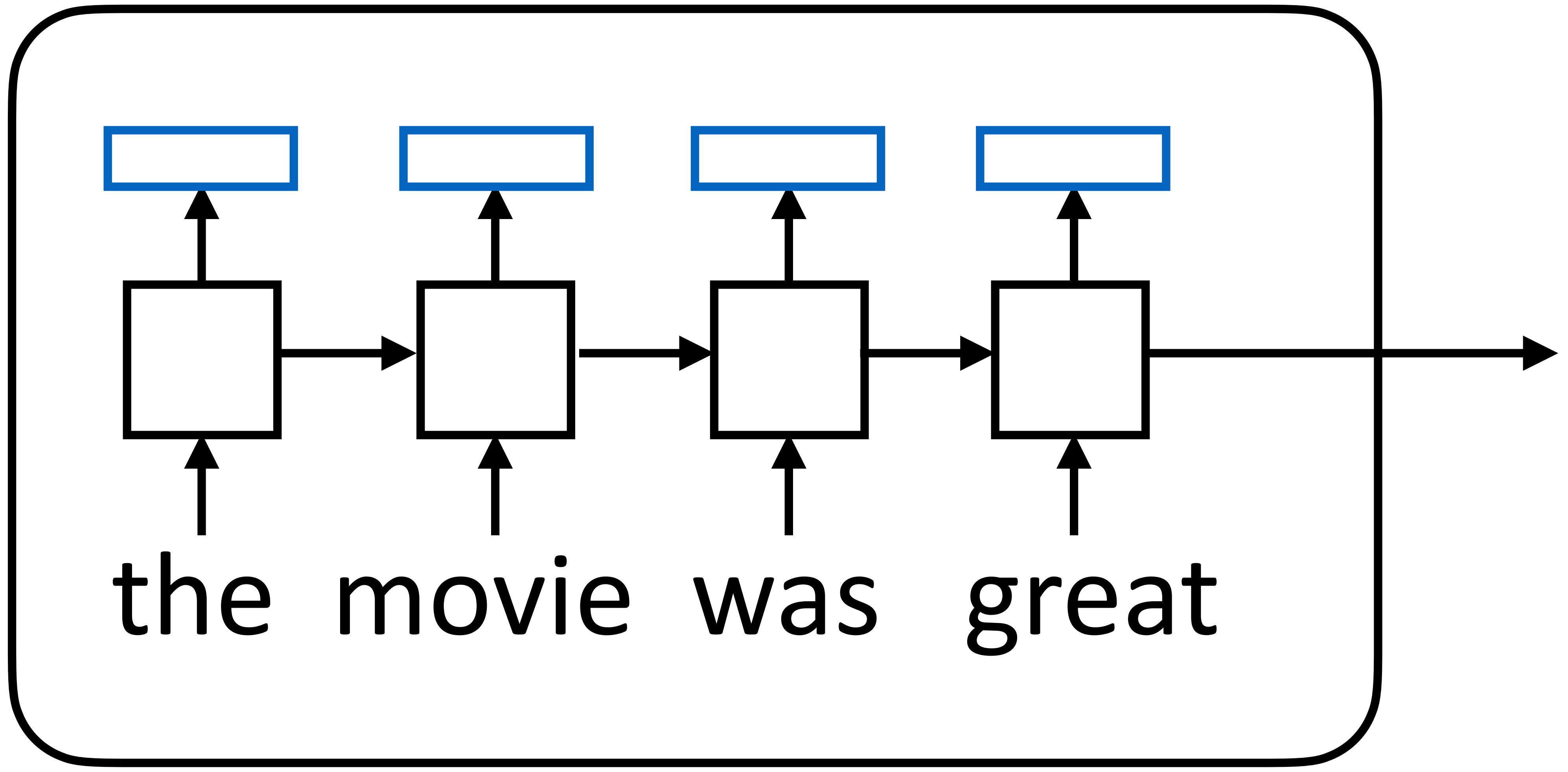


Attention

- Encoder hidden states capture contextual source word identity
- Decoder hidden states are now mostly responsible for selecting what to attend to
- Doesn't take a complex hidden state to walk monotonically through a sentence and spit out word-by-word translations



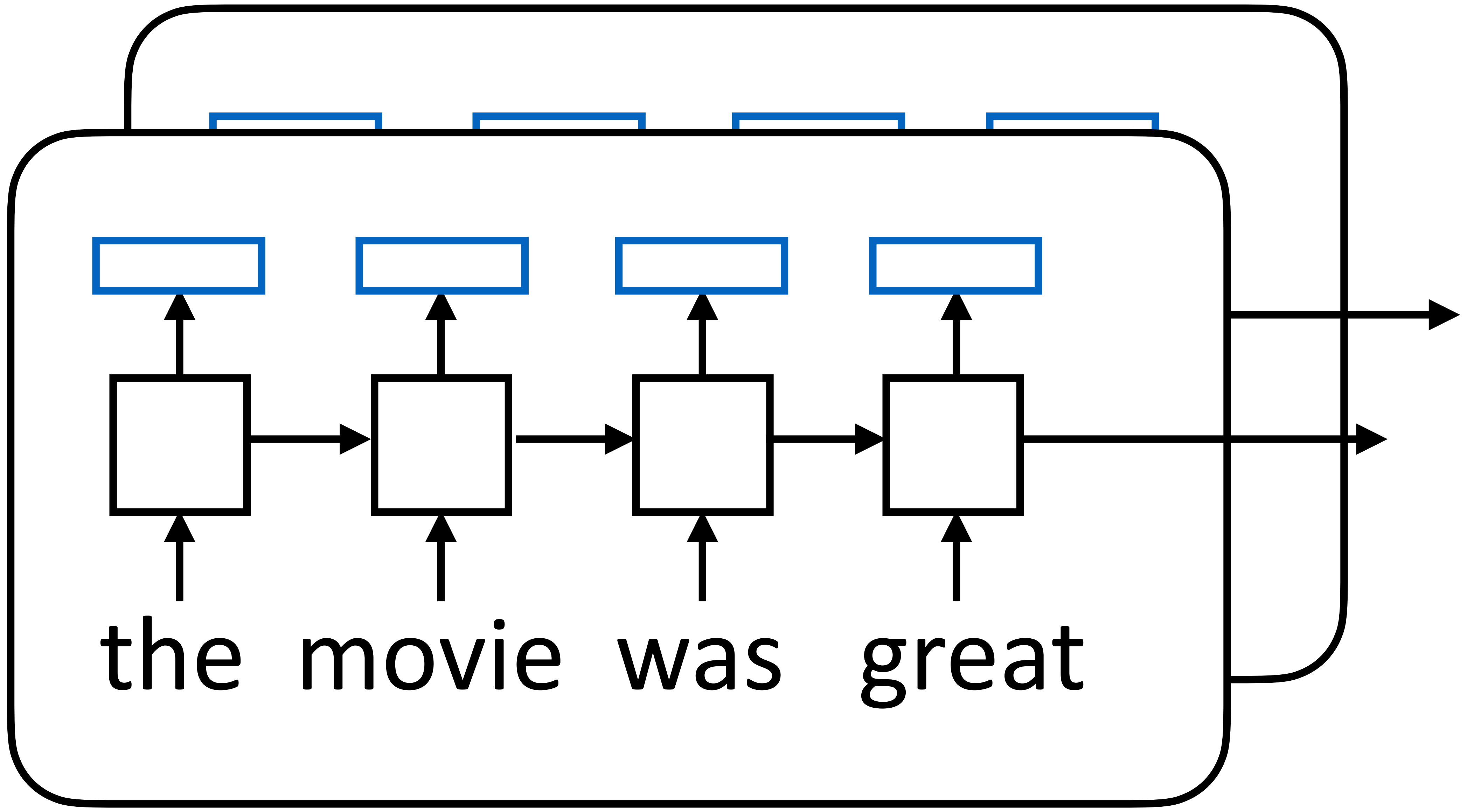
Batching Attention



Luong et al. (2015)

Batching Attention

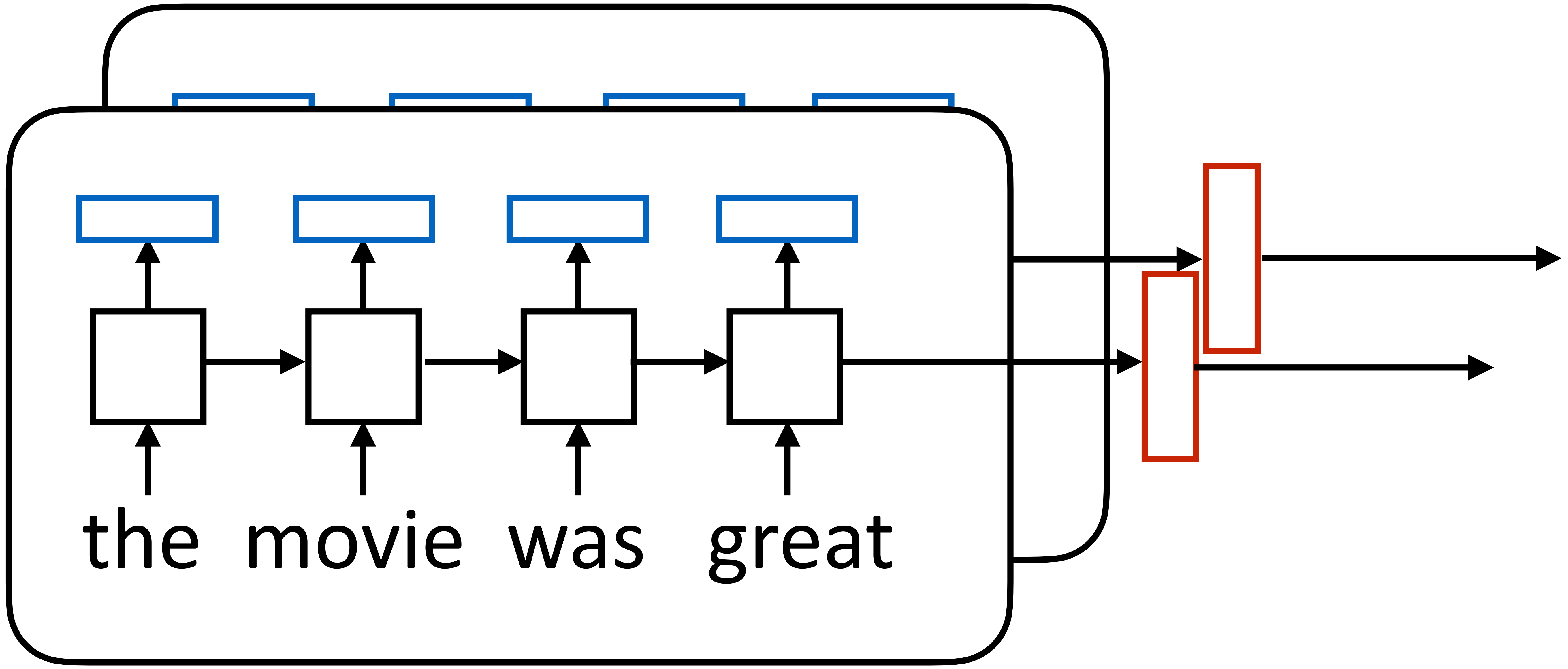
token outputs: batch size x sentence length x dimension



Luong et al. (2015)

Batching Attention

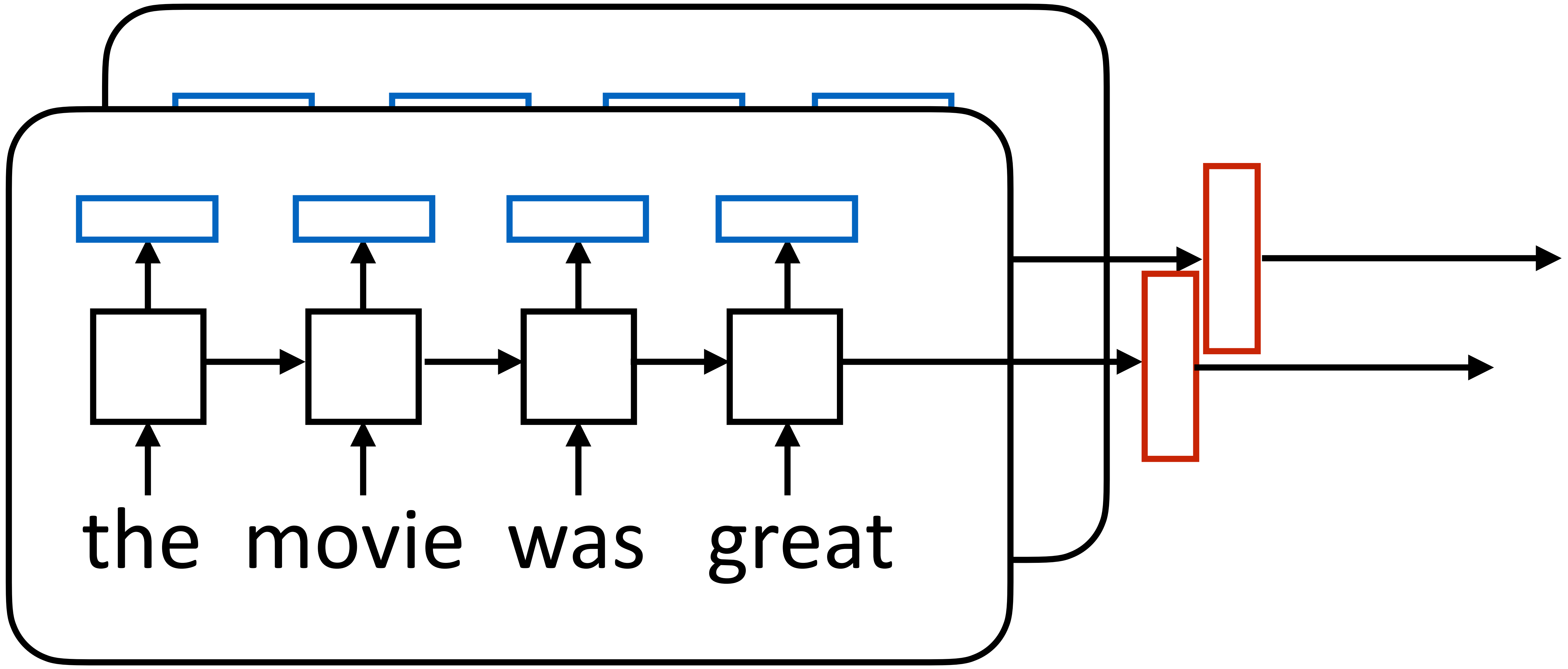
token outputs: batch size x sentence length x dimension



Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension

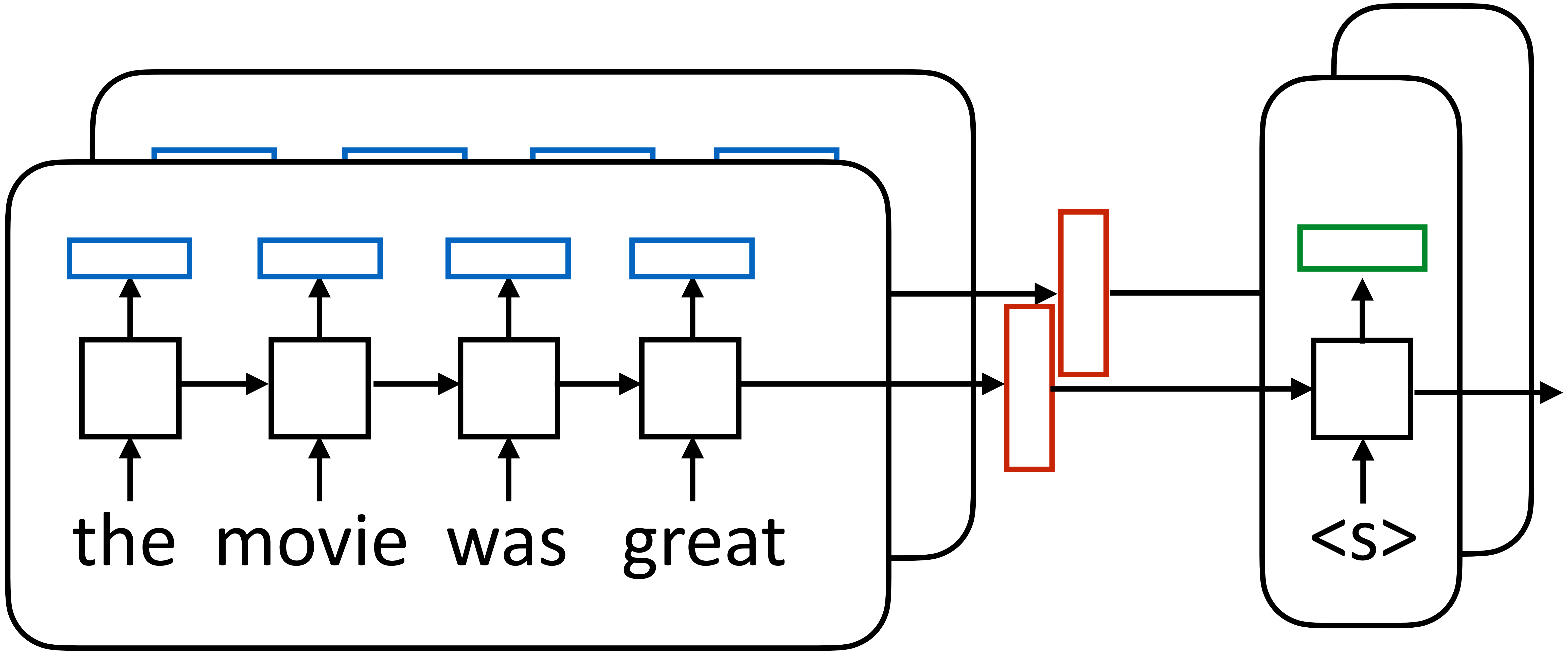


sentence outputs:
batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension

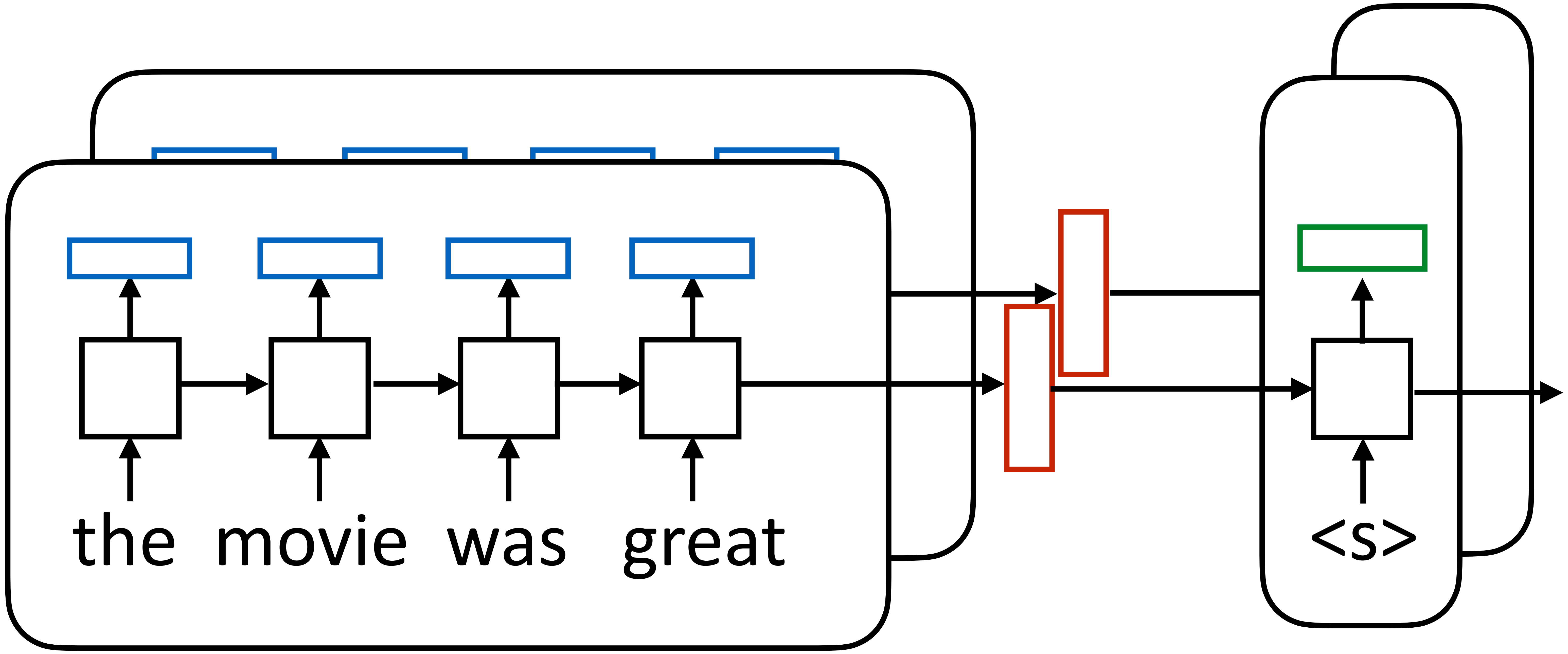


sentence outputs:
batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



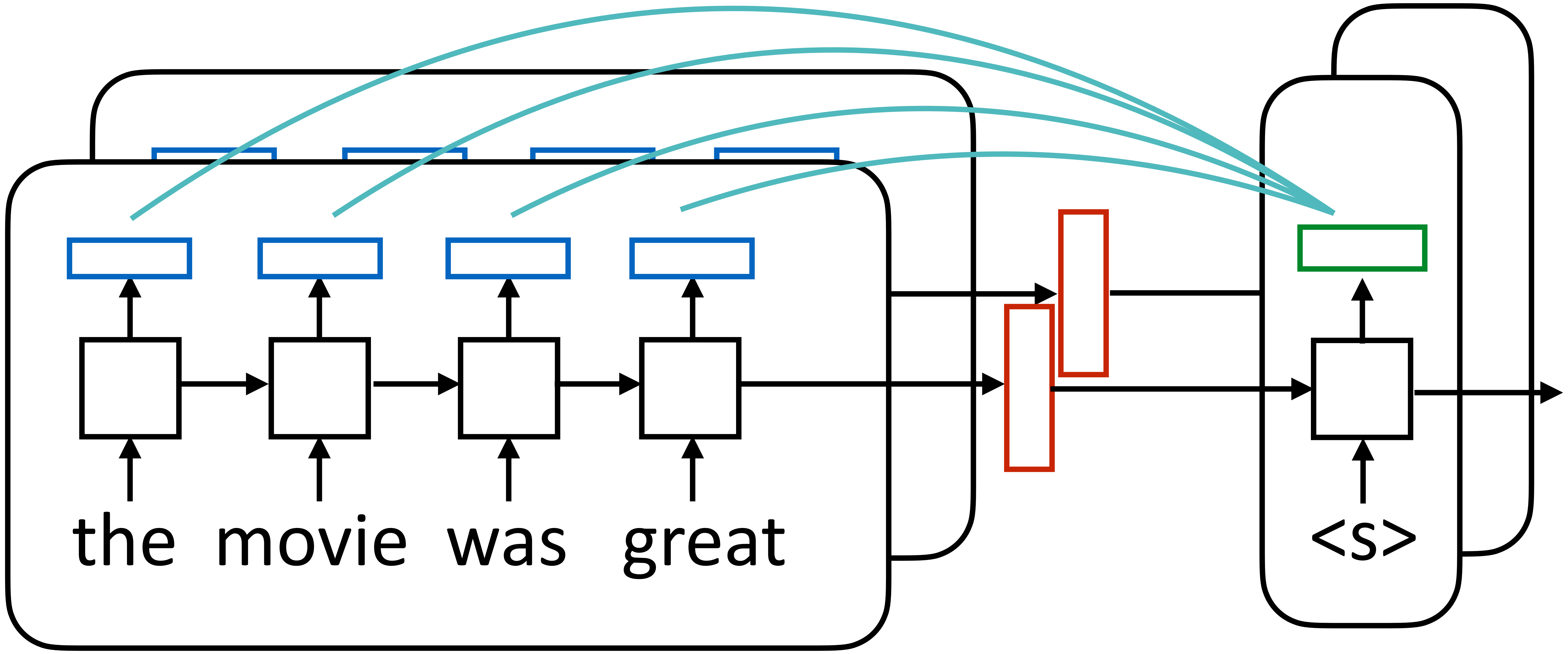
hidden state: batch size x hidden size

sentence outputs: batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



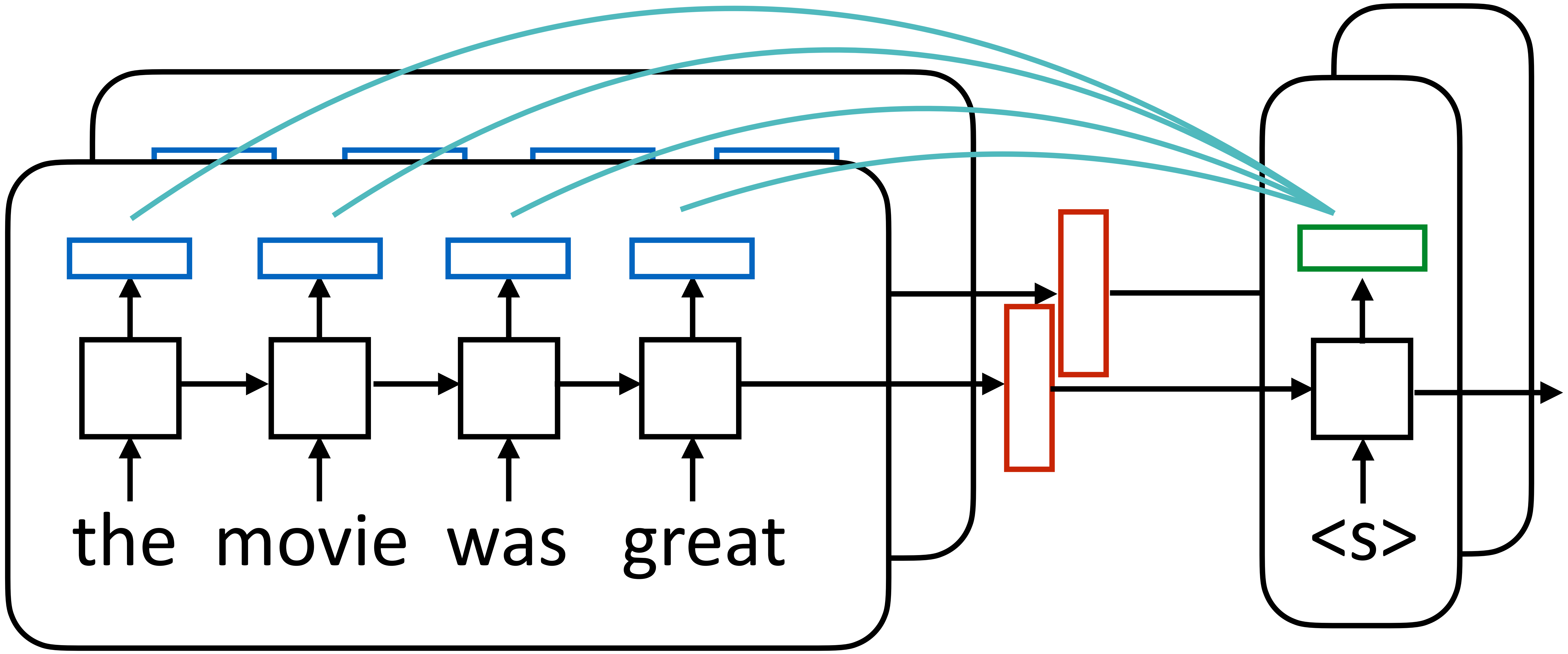
hidden state: batch size x hidden size

sentence outputs: batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



hidden state: batch size x hidden size

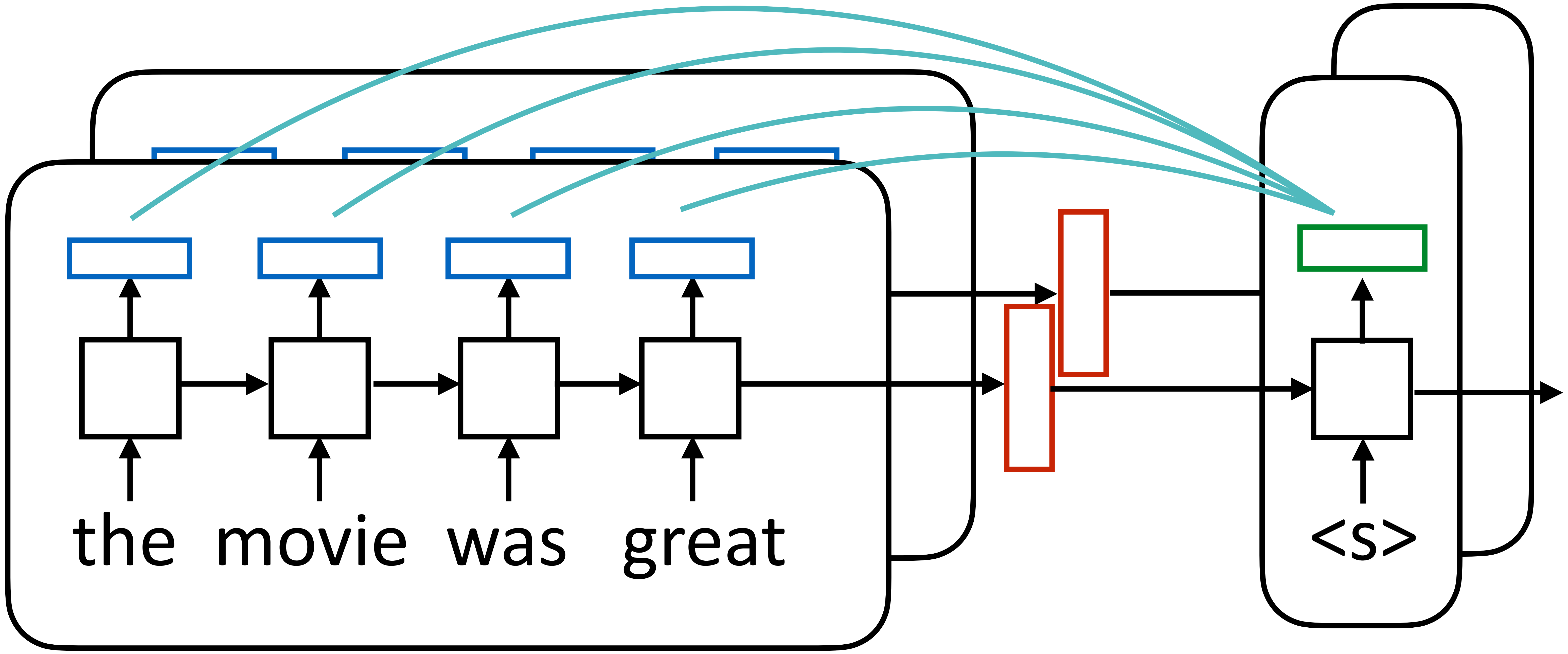
$$e_{ij} = f(\bar{h}_i, h_j)$$

sentence outputs:
batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



hidden state: batch size x hidden size

$$e_{ij} = f(\bar{h}_i, h_j)$$

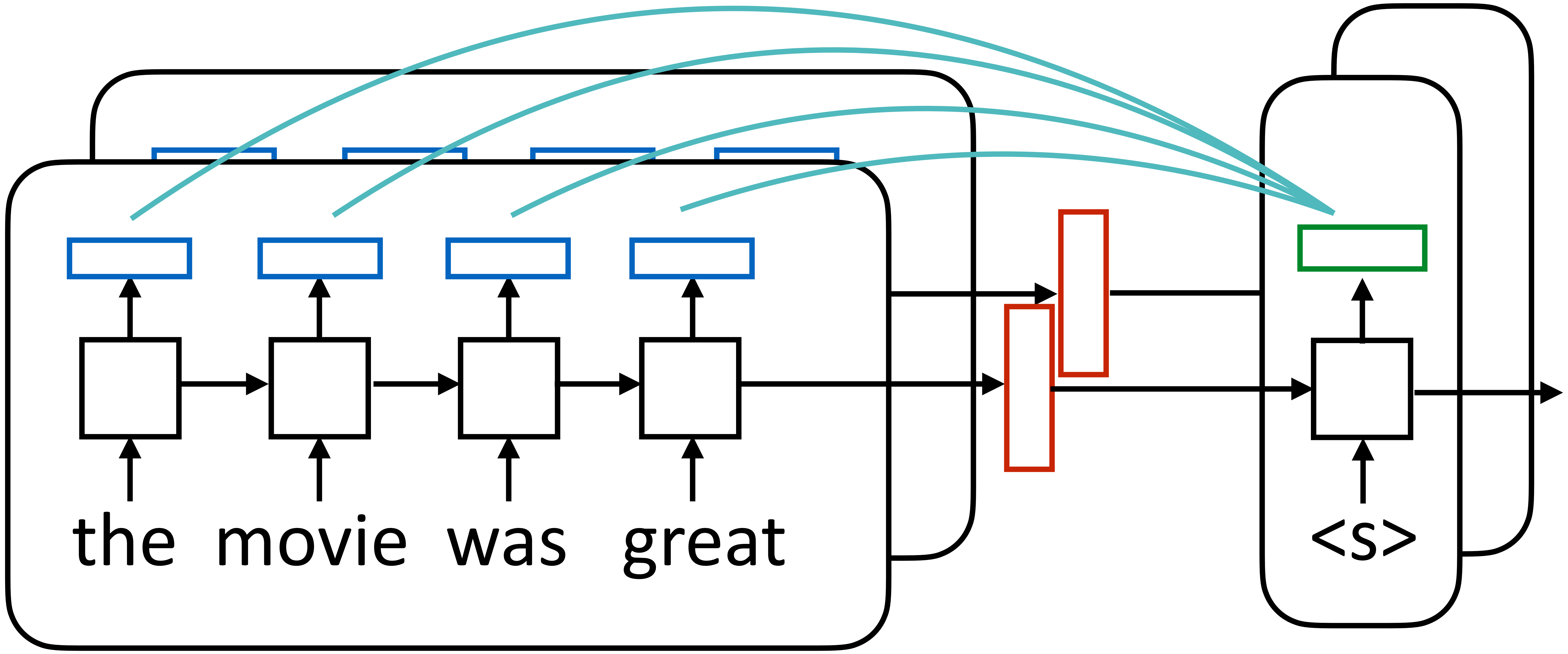
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

sentence outputs:
batch size x hidden size

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



hidden state: batch size x hidden size

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

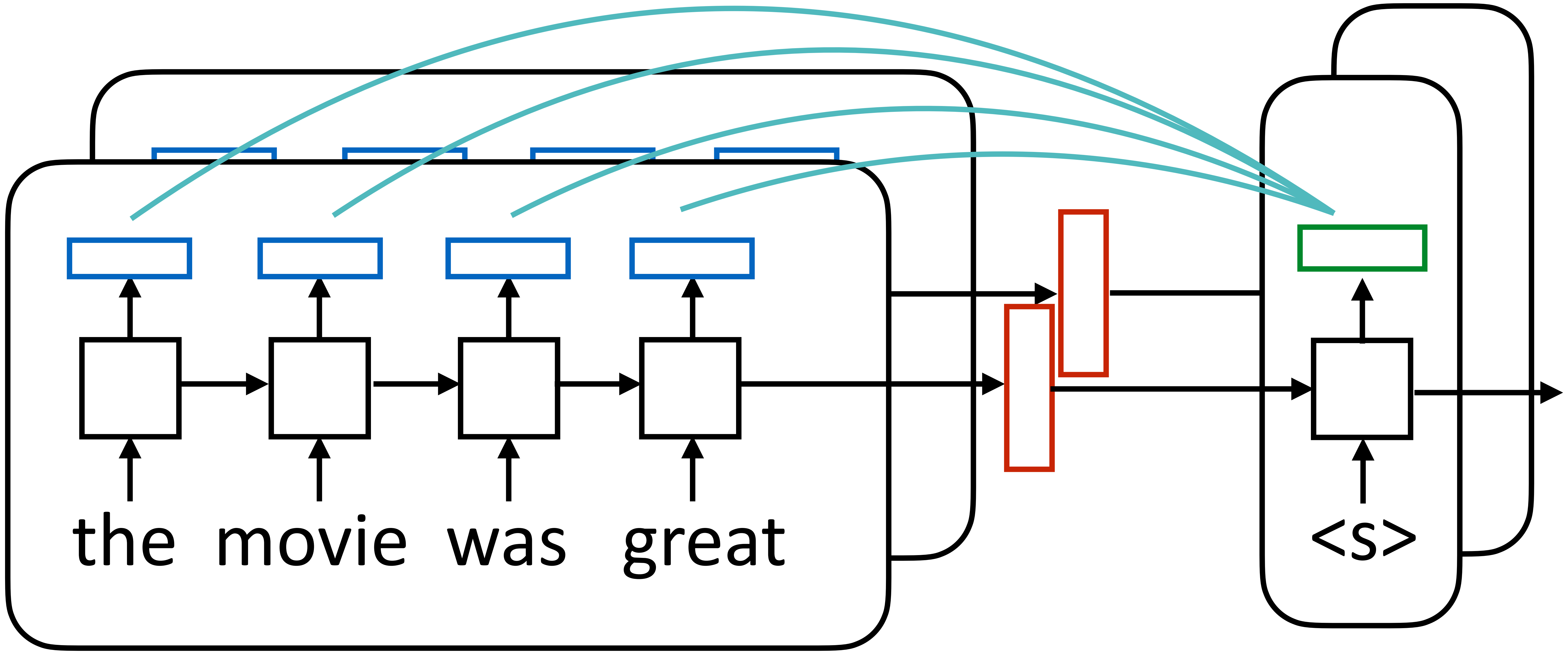
sentence outputs: batch size x hidden size

attention scores = batch size x sentence length

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



hidden state: batch size x hidden size

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

sentence outputs:
batch size x hidden size

attention scores = batch size x sentence length

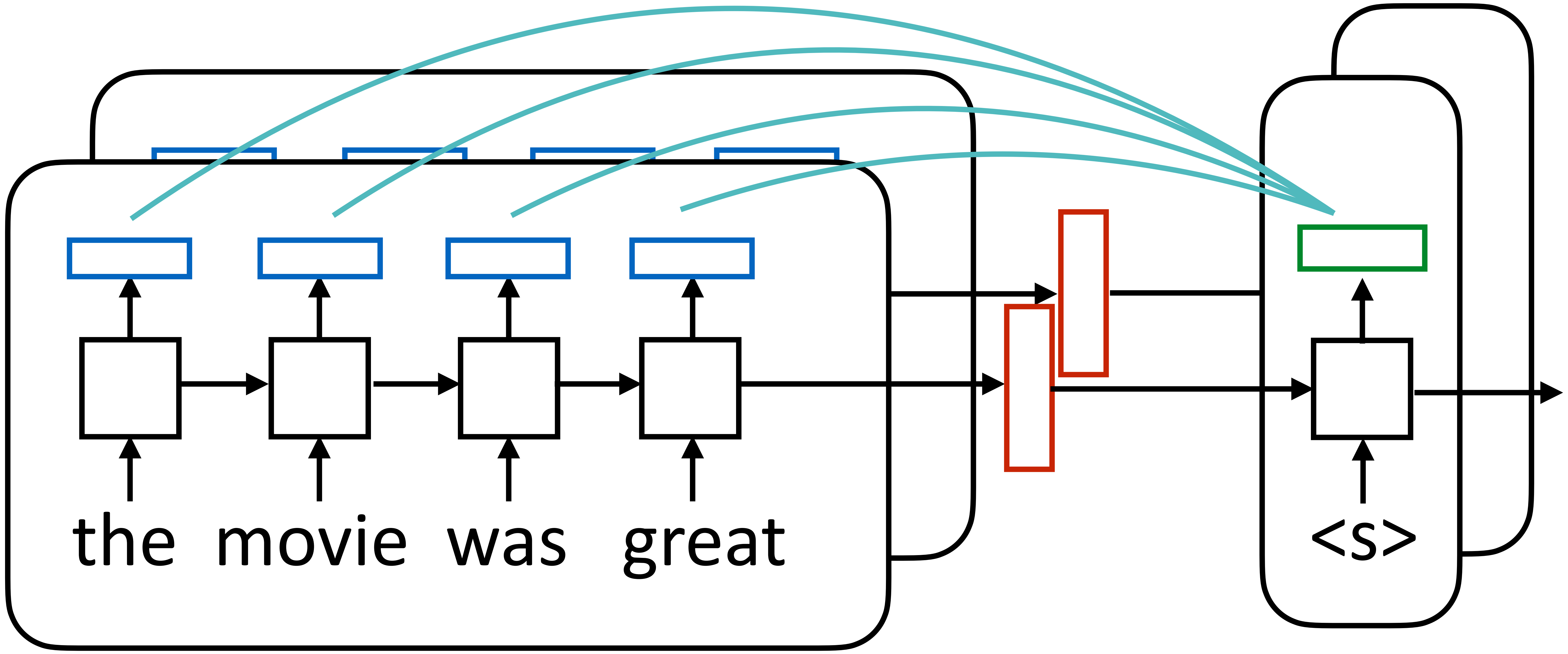
c = batch size x hidden size

$$c_i = \sum_j \alpha_{ij} h_j$$

Luong et al. (2015)

Batching Attention

token outputs: batch size x sentence length x dimension



hidden state: batch size x hidden size

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

sentence outputs:
batch size x hidden size

attention scores = batch size x sentence length

c = batch size x hidden size

$$c_i = \sum_j \alpha_{ij} h_j$$

- Make sure tensors are the right size!

Luong et al. (2015)

Results

Luong et al. (2015)
Chopra et al. (2016)
Jia and Liang (2016)

Results

- Machine translation: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with smarter attention (we'll come back to this later)

Luong et al. (2015)
Chopra et al. (2016)
Jia and Liang (2016)

Results

- Machine translation: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with smarter attention (we'll come back to this later)

- Summarization/headline generation: bigram recall from 11% -> 15%

Luong et al. (2015)
Chopra et al. (2016)
Jia and Liang (2016)

Results

- Machine translation: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with smarter attention (we'll come back to this later)

- Summarization/headline generation: bigram recall from 11% -> 15%

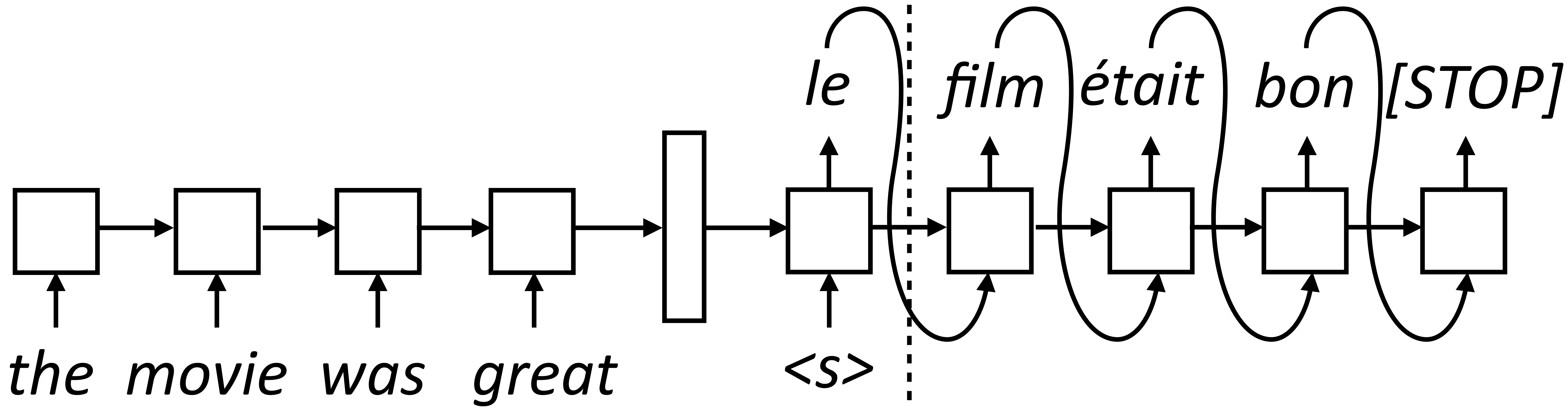
- Semantic parsing: ~30% accuracy -> 70+% accuracy on Geoquery

Luong et al. (2015)
Chopra et al. (2016)
Jia and Liang (2016)

Decoding Strategies

Greedy Decoding

- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state. This is **greedy decoding**

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$

(or attention/copying/etc.)

$$y_{\mathrm{pred}} = \mathrm{argmax}_y P(y|\mathbf{x}, y_1, \ldots, y_{i-1})$$

Problems with Greedy Decoding

- Only returns one solution, and it may not be optimal
- Can address this with **beam search**, which usually works better…but even beam search may not find the correct answer! (max probability sequence)

Model	**Beam-10**	
	BLEU	**#Search err.**
LSTM*	28.6	58.4%
SliceNet*	28.8	46.0%
Transformer-Base	30.3	57.7%
Transformer-Big*	31.7	32.1%

Stahlberg and Byrne (2019)

"Problems" with Beam Decoding

- For machine translation, the highest probability sequence is often the empty string! (>50% of the time)

Search	BLEU	Ratio	#Search errors	#Empty
Greedy	29.3	1.02	73.6%	0.0%
Beam-10	30.3	1.00	57.7%	0.0%
Exact	2.1	0.06	0.0%	51.8%

- Beam search results in *fortuitous search errors* that avoid these bad solutions

Stahlberg and Byrne (2019)

Sampling

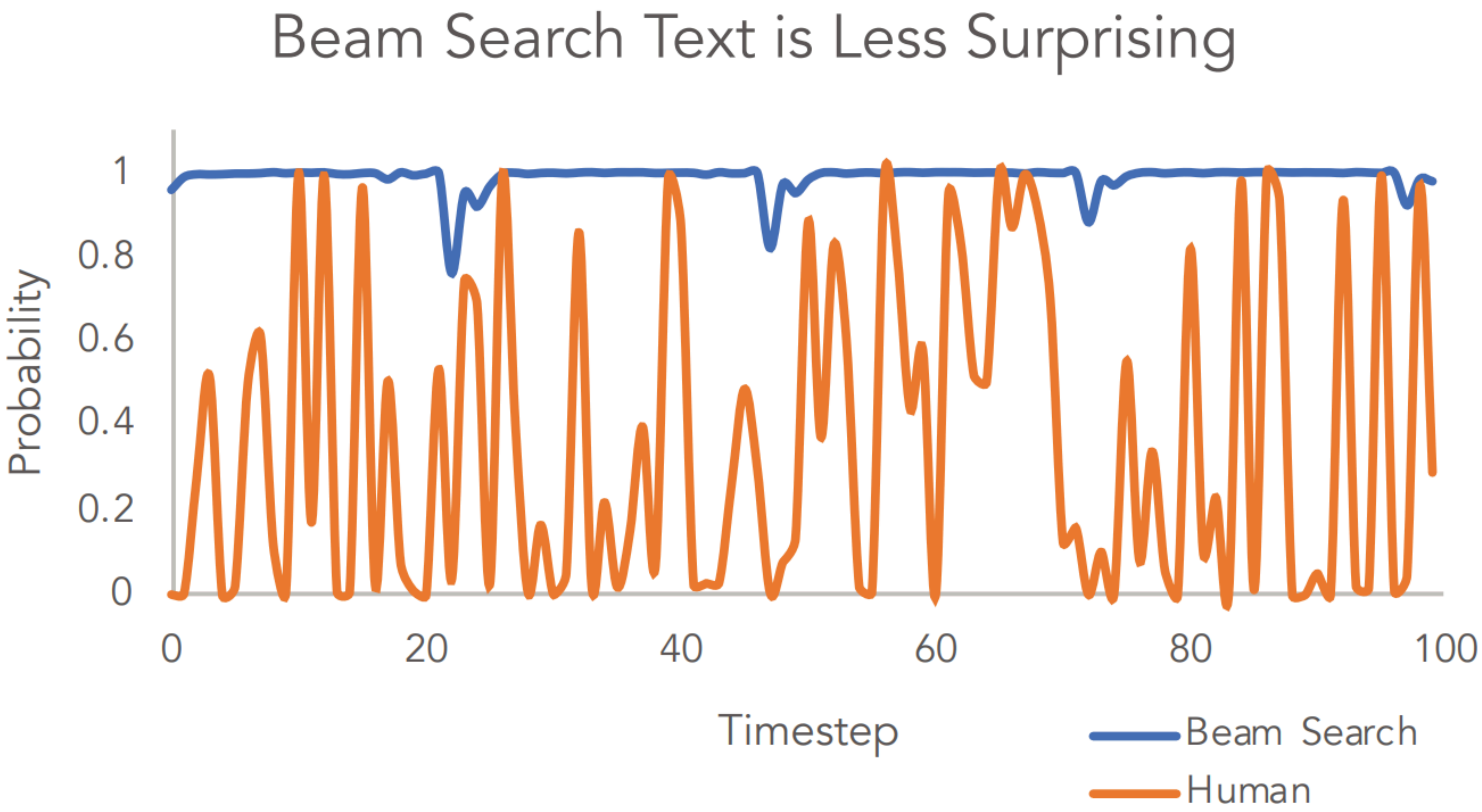
- Beam search may give many similar sequences, and these actually may be *too close* to the optimal. Can sample instead:

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$

$$y_{\mathrm{sampled}} \sim P(y|\mathbf{x}, y_1, \ldots, y_{i-1})$$

- Text *degeneration*: greedy solution can be uninteresting / vacuous for various reasons. Sampling can help.

Beam Search vs. Sampling



Holtzman et al. (2019)

Beam Search vs. Sampling

- These are samples from an unconditioned language model (not seq2seq model)

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

- Sampling is better but sometimes draws too far from the tail of the distribution

Holtzman et al. (2019)

Decoding Strategies

- Greedy
- Beam search
- Sampling
- Nucleus or top-k sampling:
  - Nucleus: take the top p% (95%) of the distribution, sample from within that
  - Top-k: take the top k most likely words (k=5), sample from those

Generation Tasks

WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Beam Search, *b*=16

The number of stranded whales has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year. The number of whales stranded on the West Australian coast has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year.

Pure Sampling

The Australian Food Safety Authority has warned Australia's beaches may be revitalised this year because healthy seabirds and seals have been on the move. More than 50,000 seabirds, sea mammals and seahorses have been swept into the sea by the Holden CS118 and Adelaide Airport CS300 from 2013. A major white-bat and umidauda migration across Australia is under way in Australia for the first time, with numbers reaching an estimated 50,000.

Nucleus, *p*=0.95

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.

Generation Tasks

- There are a range of seq2seq modeling tasks we will address
- For more constrained problems: greedy/beam decoding are usually best
- For less constrained problems: nucleus sampling introduces favorable variation in the output

Less constrained ⟷ More constrained

Unconditioned sampling/ "story generation"

Dialogue

Summarization

Data-to-text

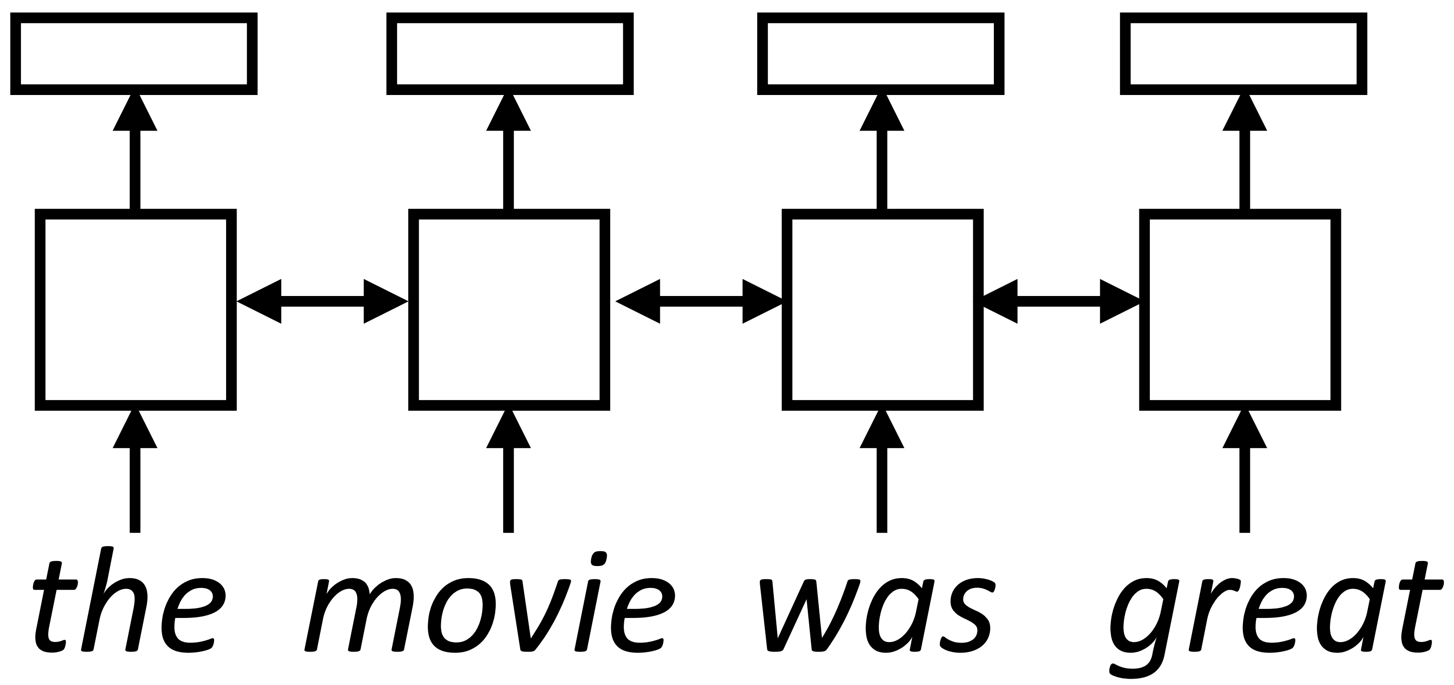
Translation

Text-to-code

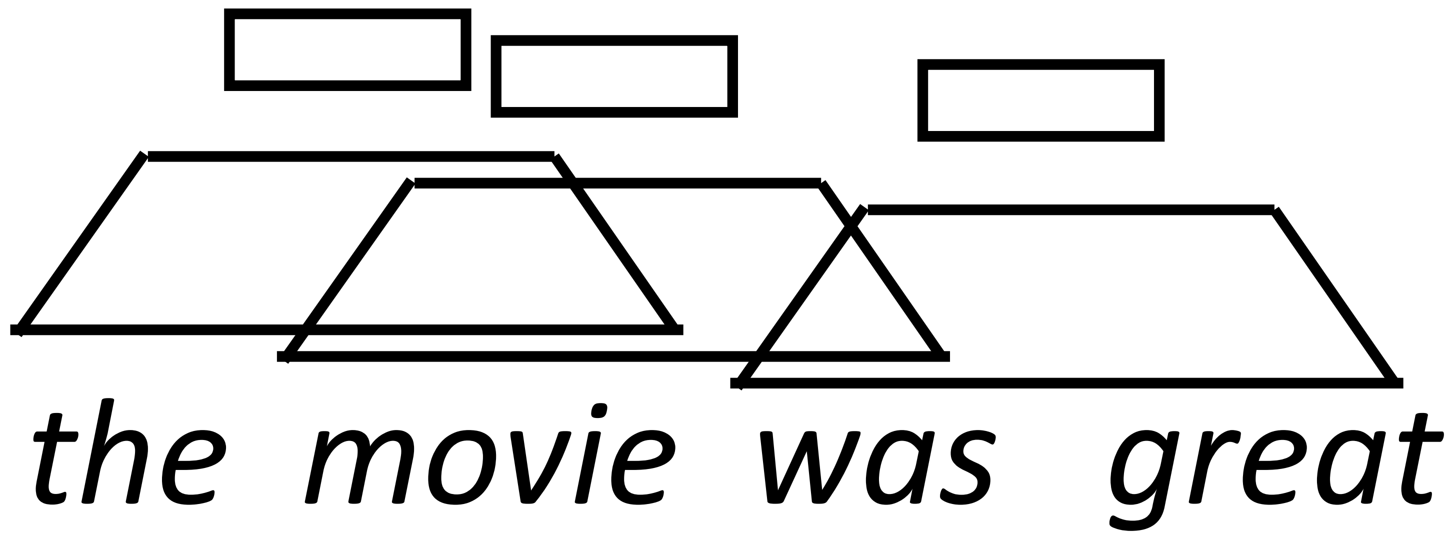
Transformers

Sentence Encoders

- LSTM abstraction: maps each vector in a sentence to a new, context-aware vector



- CNNs do something similar with filters



- Attention can give us a third way to do this

Vaswani et al. (2017)

Self-Attention

- Assume we're using GloVe — what do we want our neural network to do?

The ballerina is very excited that she will dance in the show.

- What words need to be contextualized here?
  - Pronouns need to look at antecedents
  - Ambiguous words should look at context
  - Words should look at syntactic parents/children
- Problem: LSTMs and CNNs don't do this

Vaswani et al. (2017)

Self-Attention

- Want:

The ballerina is very excited that she will dance in the show.

- LSTMs/CNNs: tend to look at local context

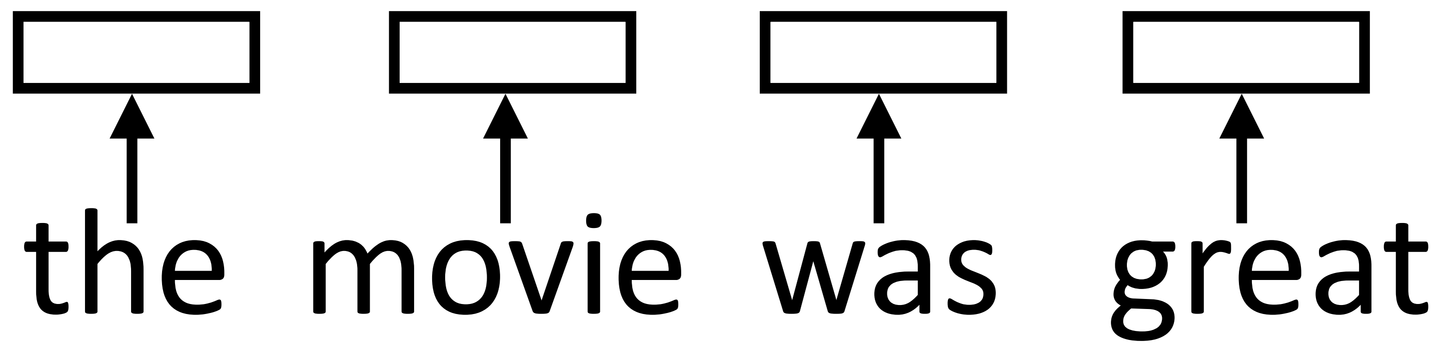
The ballerina is very excited that she will dance in the show.

- To appropriately contextualize embeddings, we need to pass information over long distances dynamically for each word

Vaswani et al. (2017)

Self-Attention

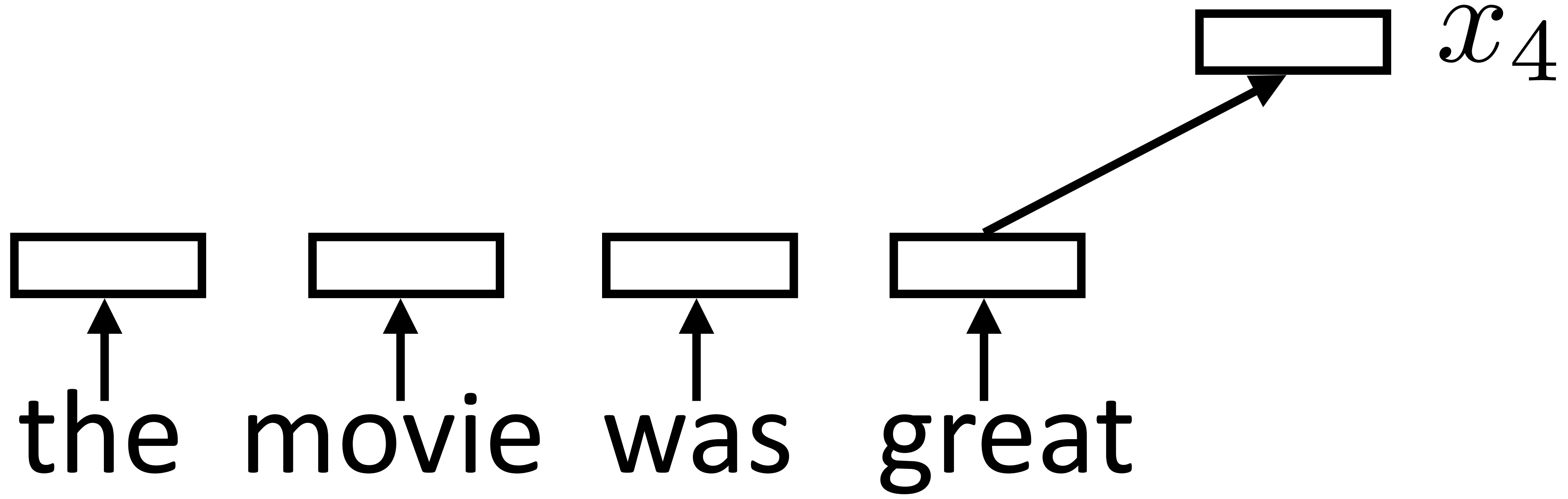
- Each word forms a "query" which then computes attention over each word



Vaswani et al. (2017)

Self-Attention

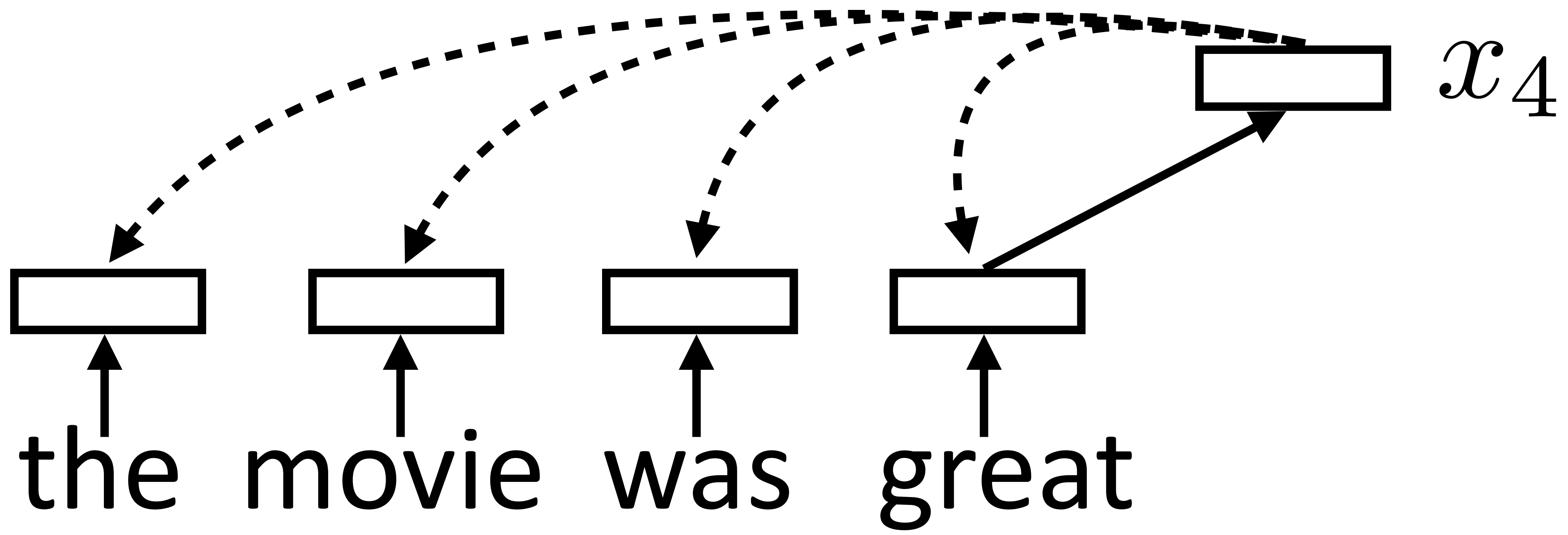
- Each word forms a "query" which then computes attention over each word



Vaswani et al. (2017)

Self-Attention

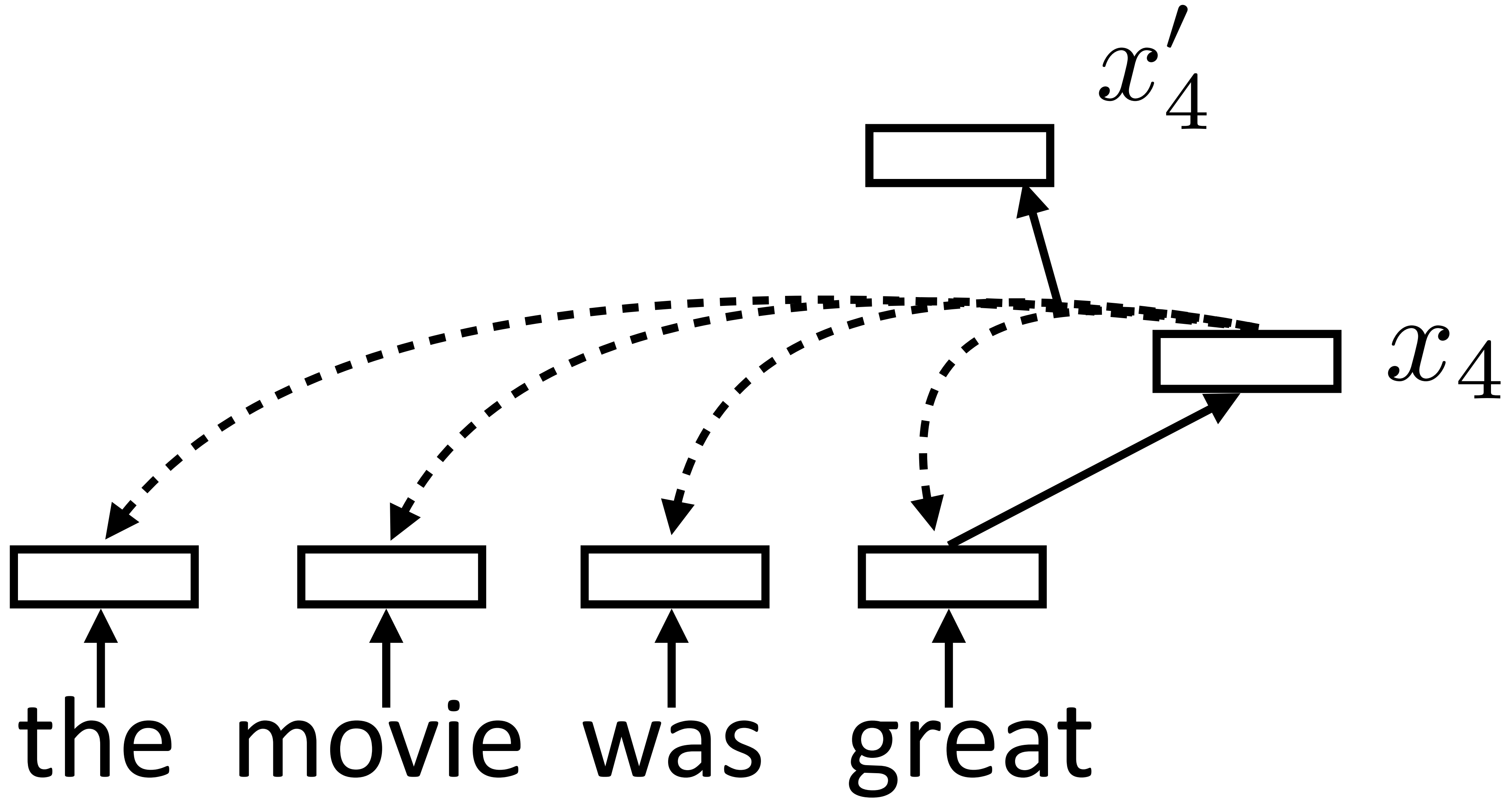
- Each word forms a "query" which then computes attention over each word



Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

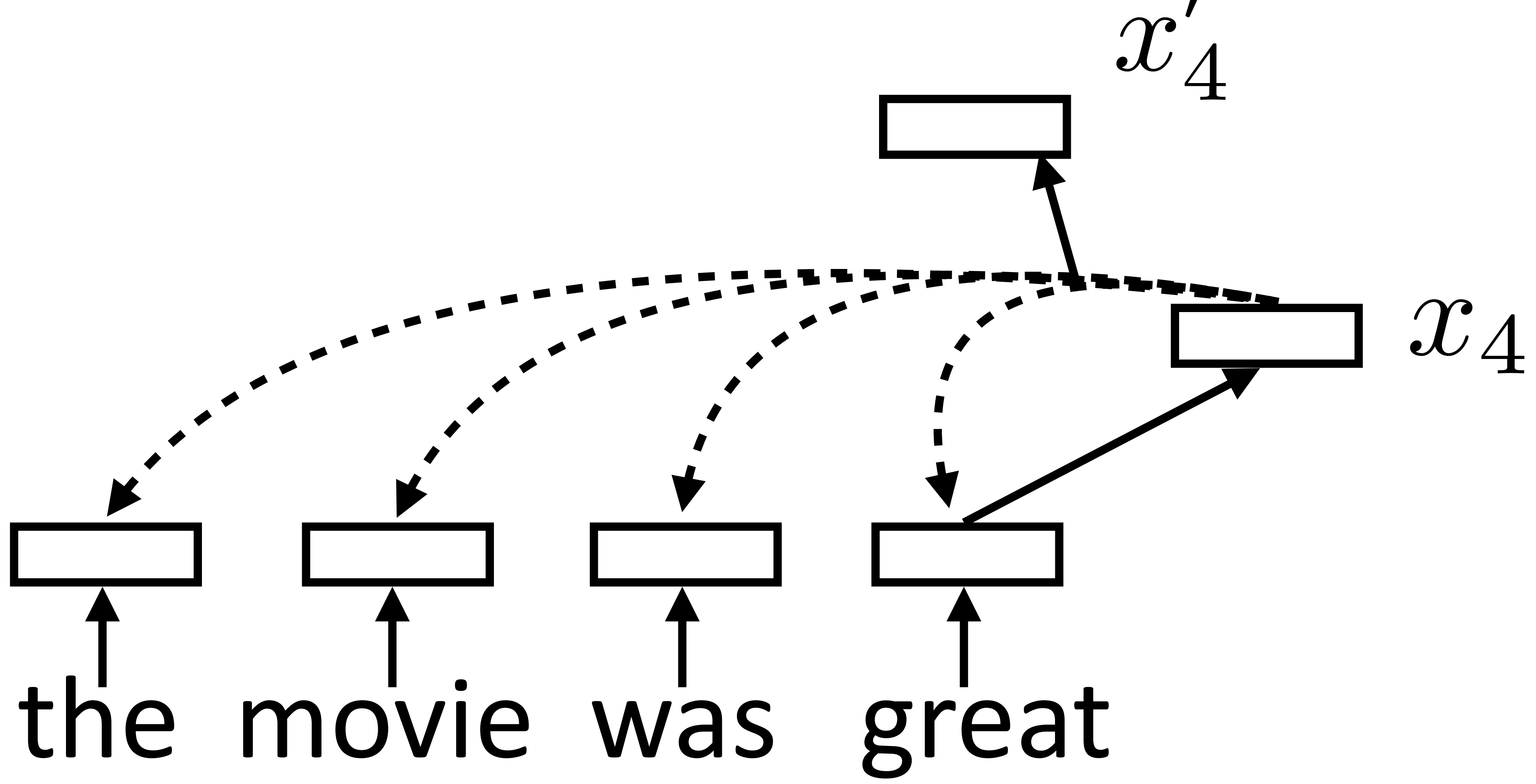


Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$



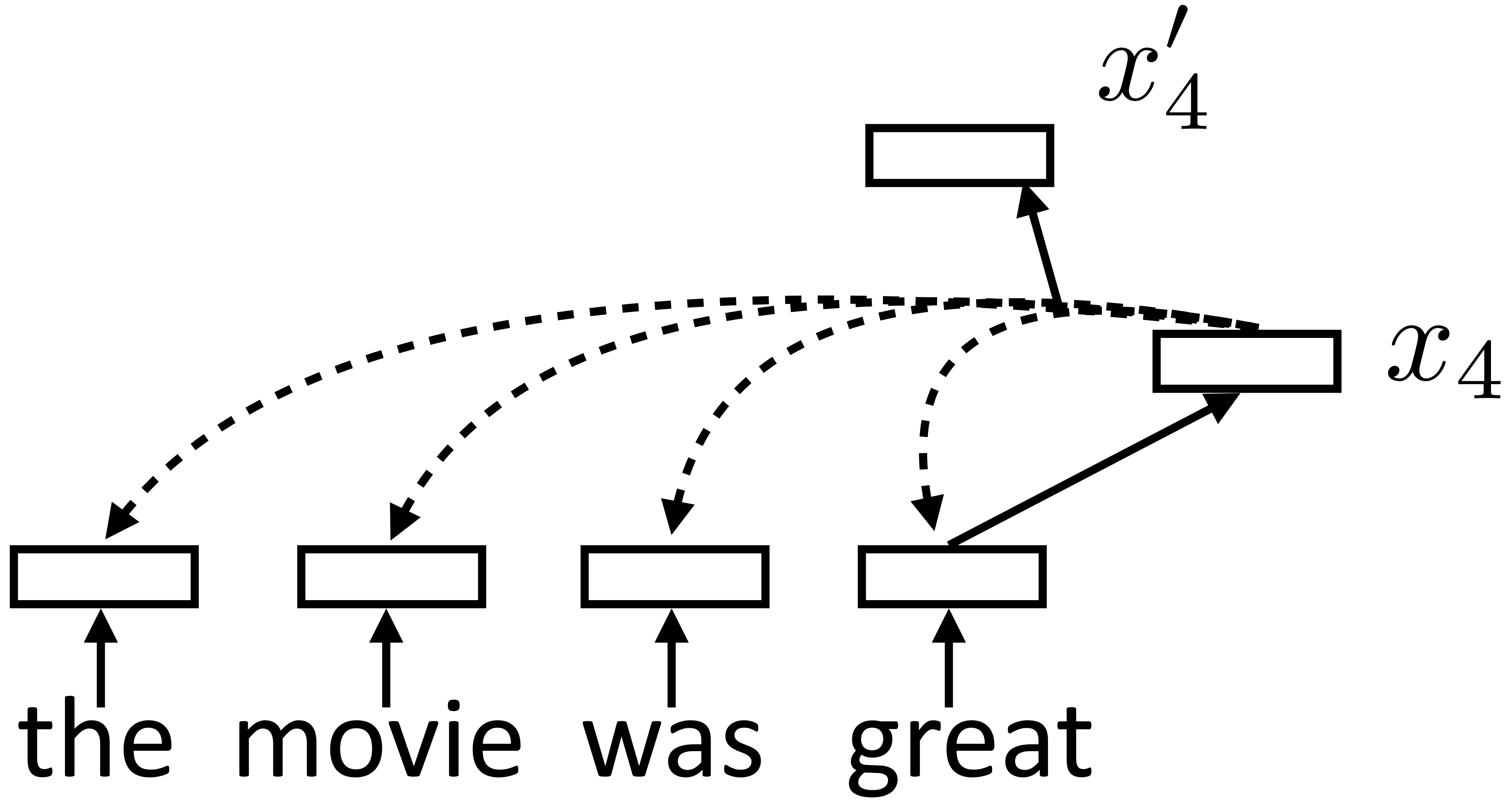
Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j)$$ scalar

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j$$ vector = sum of scalar * vector



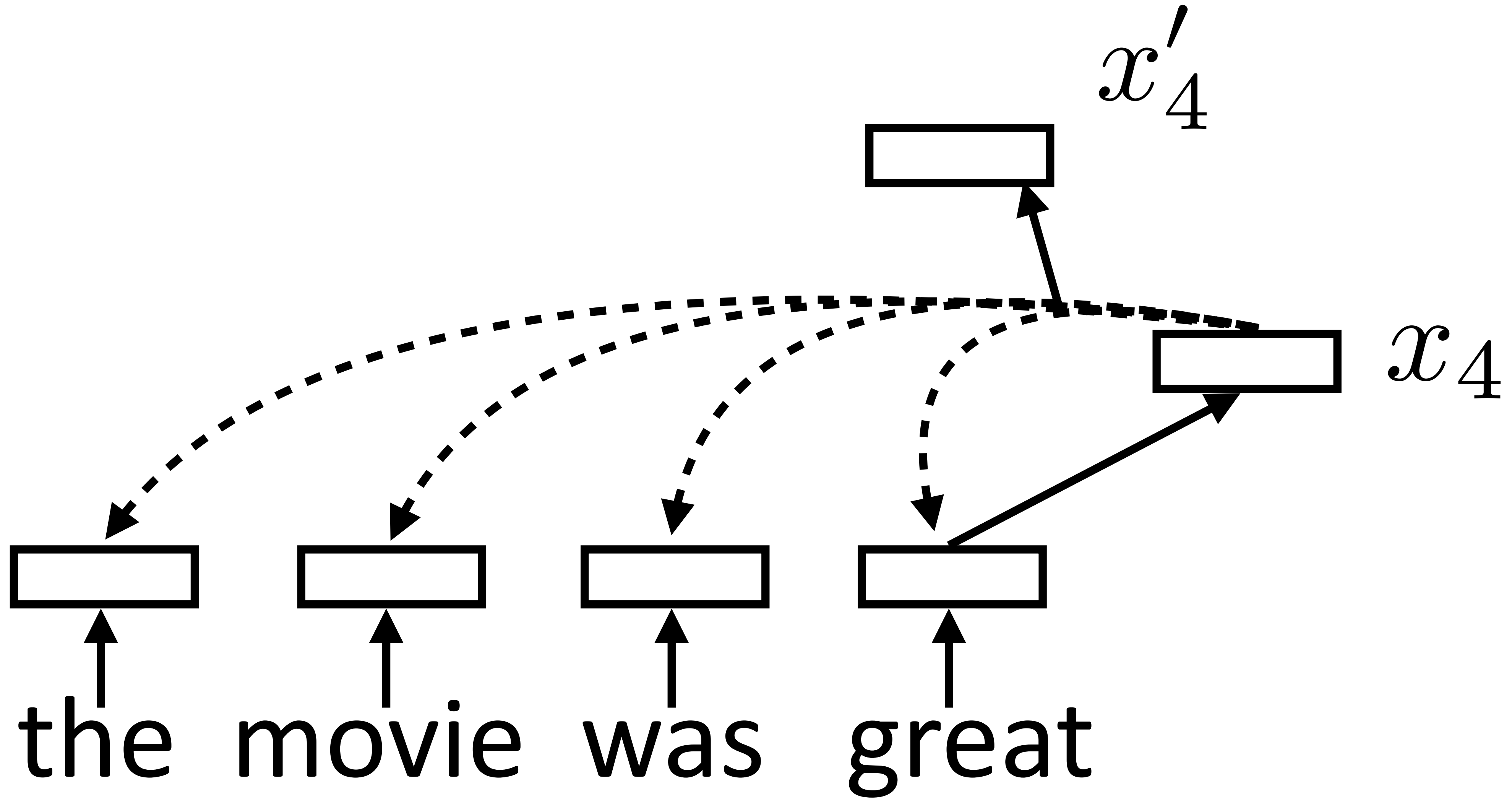
Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j)$$ scalar

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j$$ vector = sum of scalar * vector



- Multiple "heads" analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

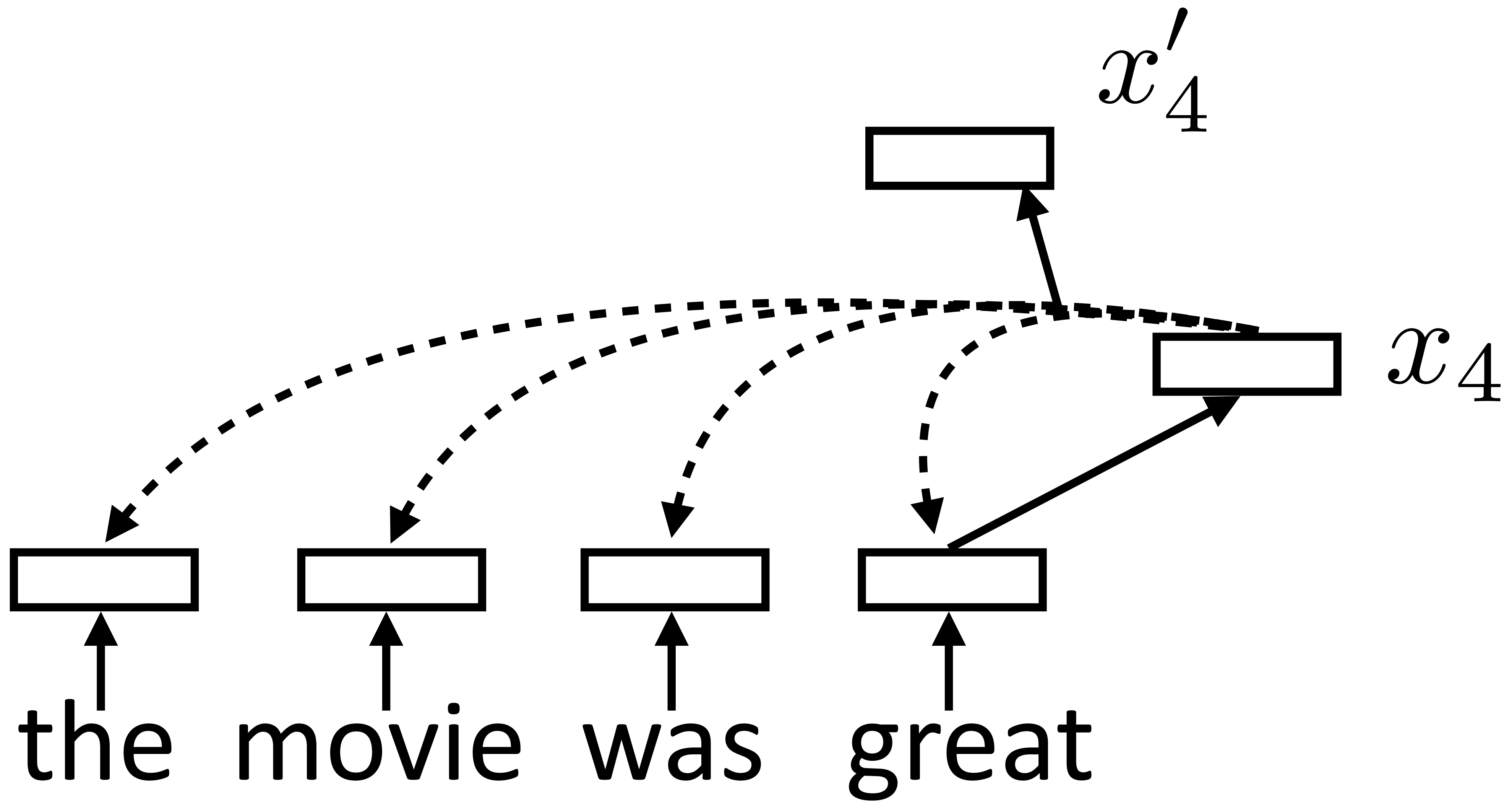
Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j \quad \text{vector = sum of scalar * vector}$$



- Multiple "heads" analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{k,i,j} = \mathrm{softmax}(x_i^\top W_k x_j)$$

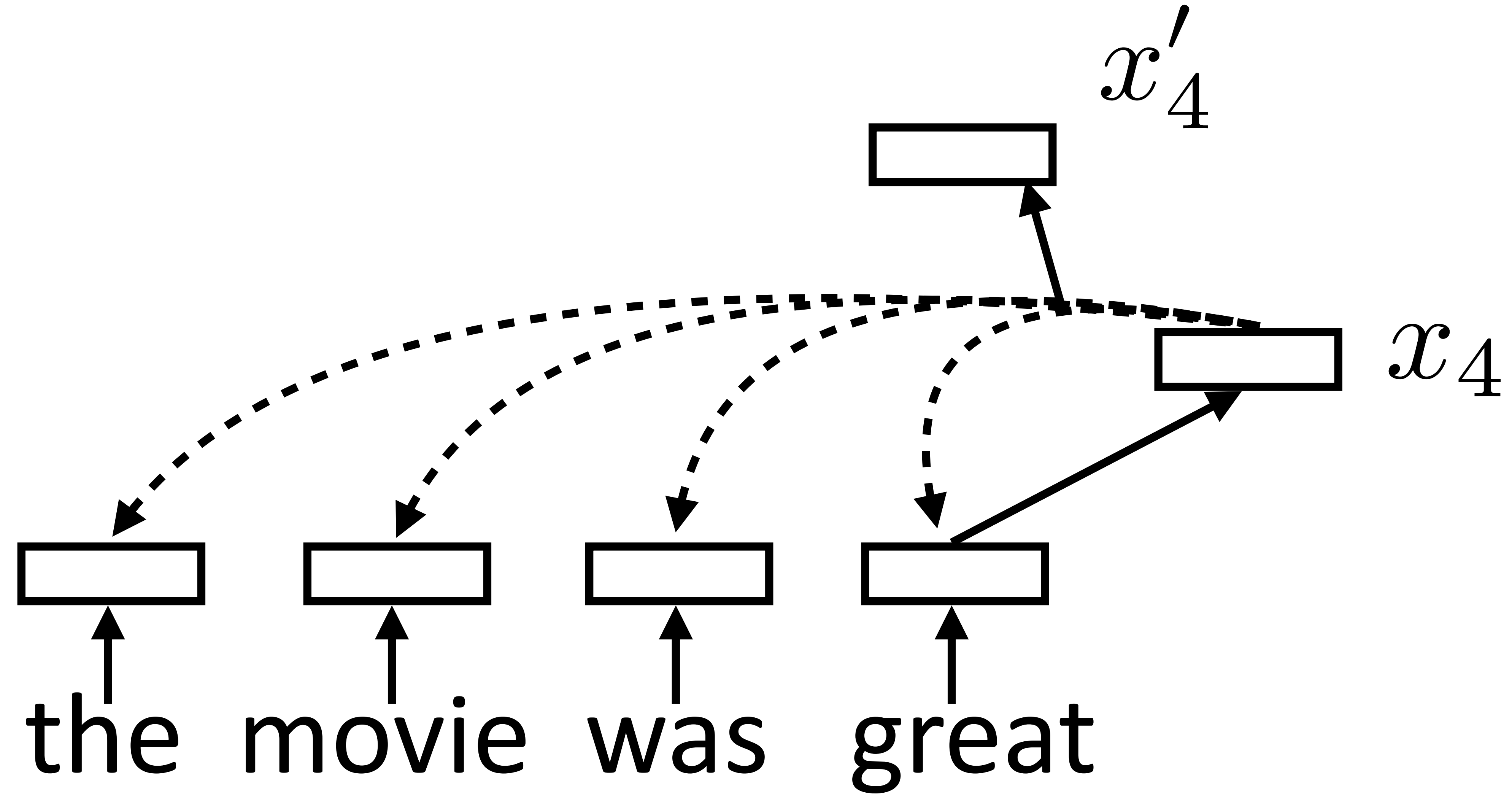
Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j \quad \text{vector = sum of scalar * vector}$$



- Multiple "heads" analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{k,i,j} = \mathrm{softmax}(x_i^\top W_k x_j) \qquad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j$$

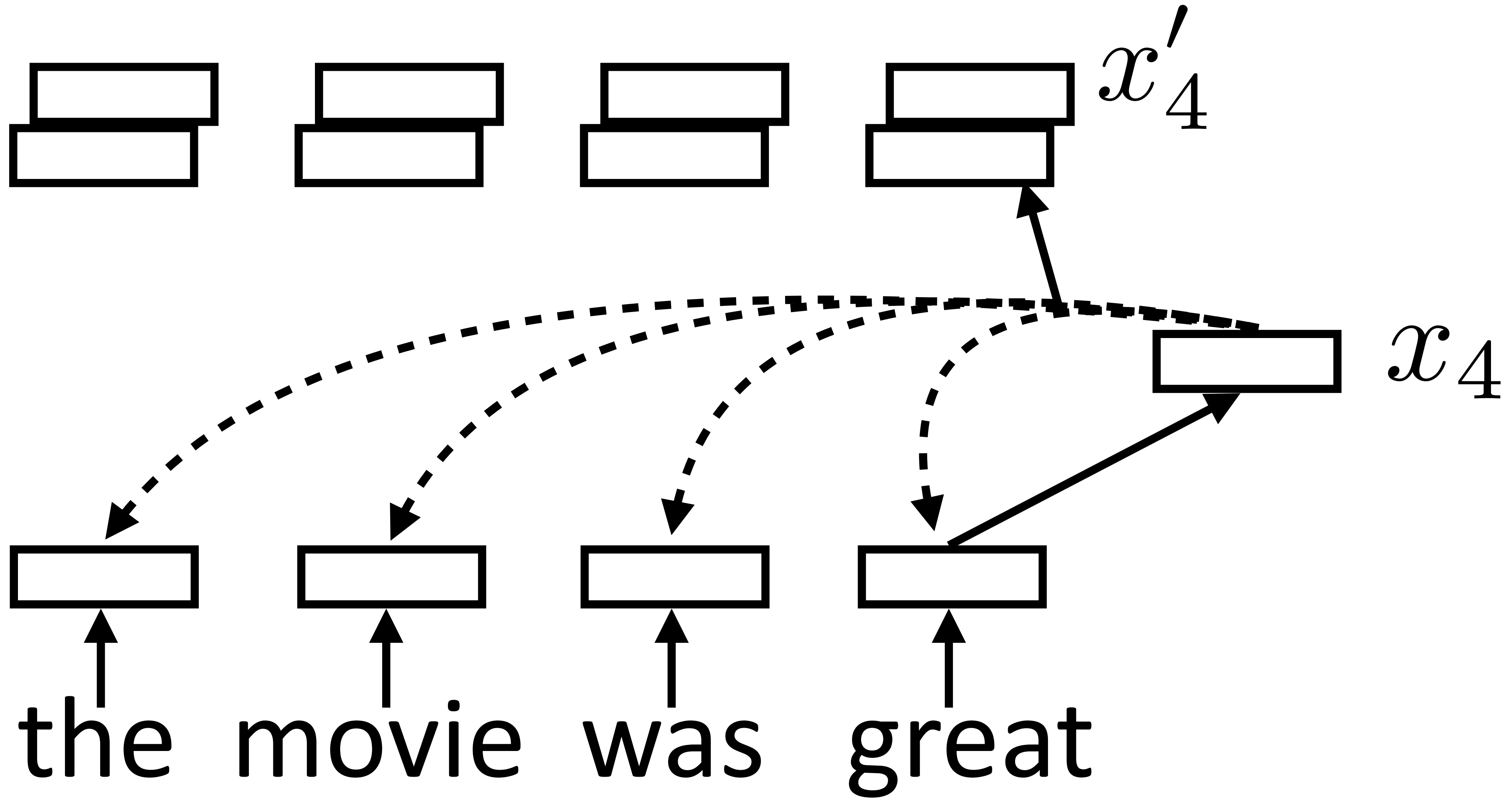
Vaswani et al. (2017)

Self-Attention

- Each word forms a "query" which then computes attention over each word

$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j \quad \text{vector = sum of scalar * vector}$$



- Multiple "heads" analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{k,i,j} = \mathrm{softmax}(x_i^\top W_k x_j) \qquad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j$$

Vaswani et al. (2017)

What can self-attention do?

The ballerina is very excited that she will dance in the show.

0	0.5	0	0	0.1	0.1	0	0.1	0.2	0	0	0

0	0.1	0	0	0	0	0	0	0.5	0	0.4	0

- Attend nearby + to semantically related terms
- This is a demonstration, we will revisit what these models actually learn when we discuss BERT
- Why multiple heads? Softmaxes end up being peaked, single distribution cannot easily put weight on multiple things

Vaswani et al. (2017)

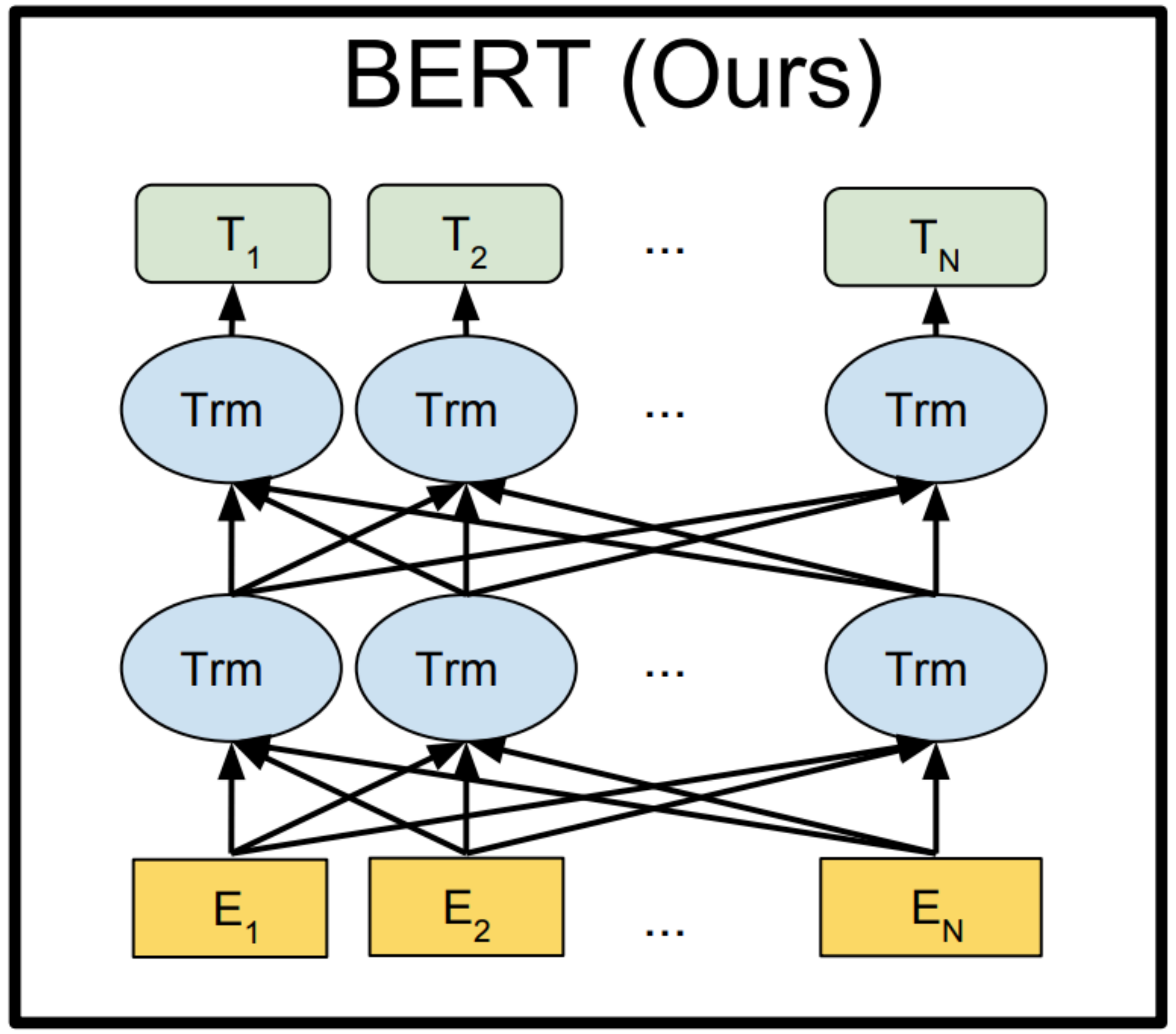
Transformer Uses

- Supervised: transformer can replace LSTM as encoder, decoder, or both; will revisit this when we discuss MT

- Unsupervised: transformers work better than LSTM for unsupervised pre-training of embeddings: predict word given context words

- BERT (Bidirectional Encoder Representations from Transformers): pretraining transformer language models similar to ELMo

- Stronger than similar methods, SOTA on ~11 tasks (including NER — 92.8 F1)



Takeaways

Takeaways

- Attention is very helpful for seq2seq models

Takeaways

- Attention is very helpful for seq2seq models
- Used for tasks including summarization and sentence ordering

Takeaways

- Attention is very helpful for seq2seq models
- Used for tasks including summarization and sentence ordering
- Explicitly copying input can be beneficial as well

Takeaways

- Attention is very helpful for seq2seq models
- Used for tasks including summarization and sentence ordering
- Explicitly copying input can be beneficial as well
- Transformers are strong models we'll come back to later