Seq2Seq + Attention

Wei Xu

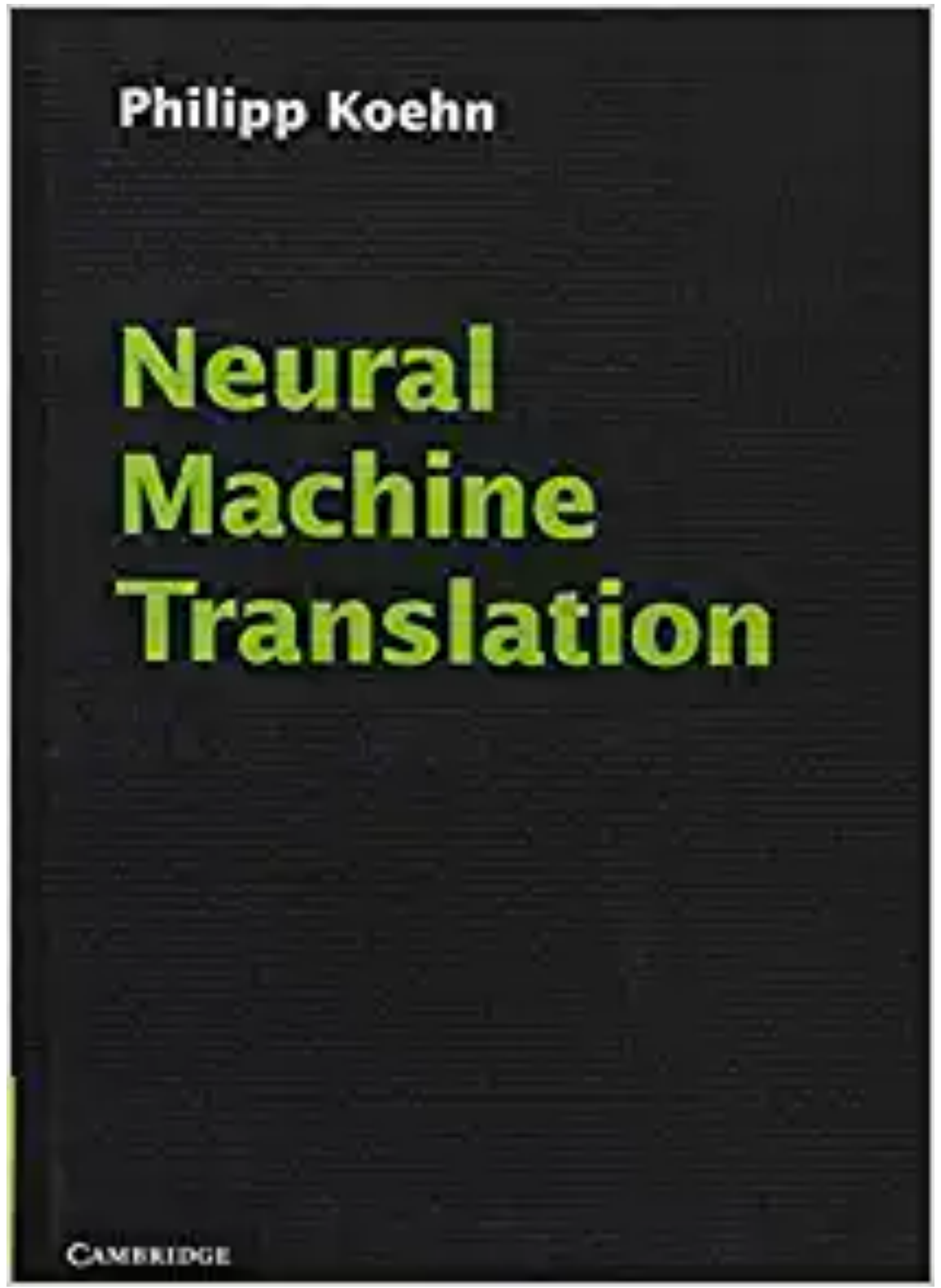
(many slides from Greg Durrett)

This & Next Lecture

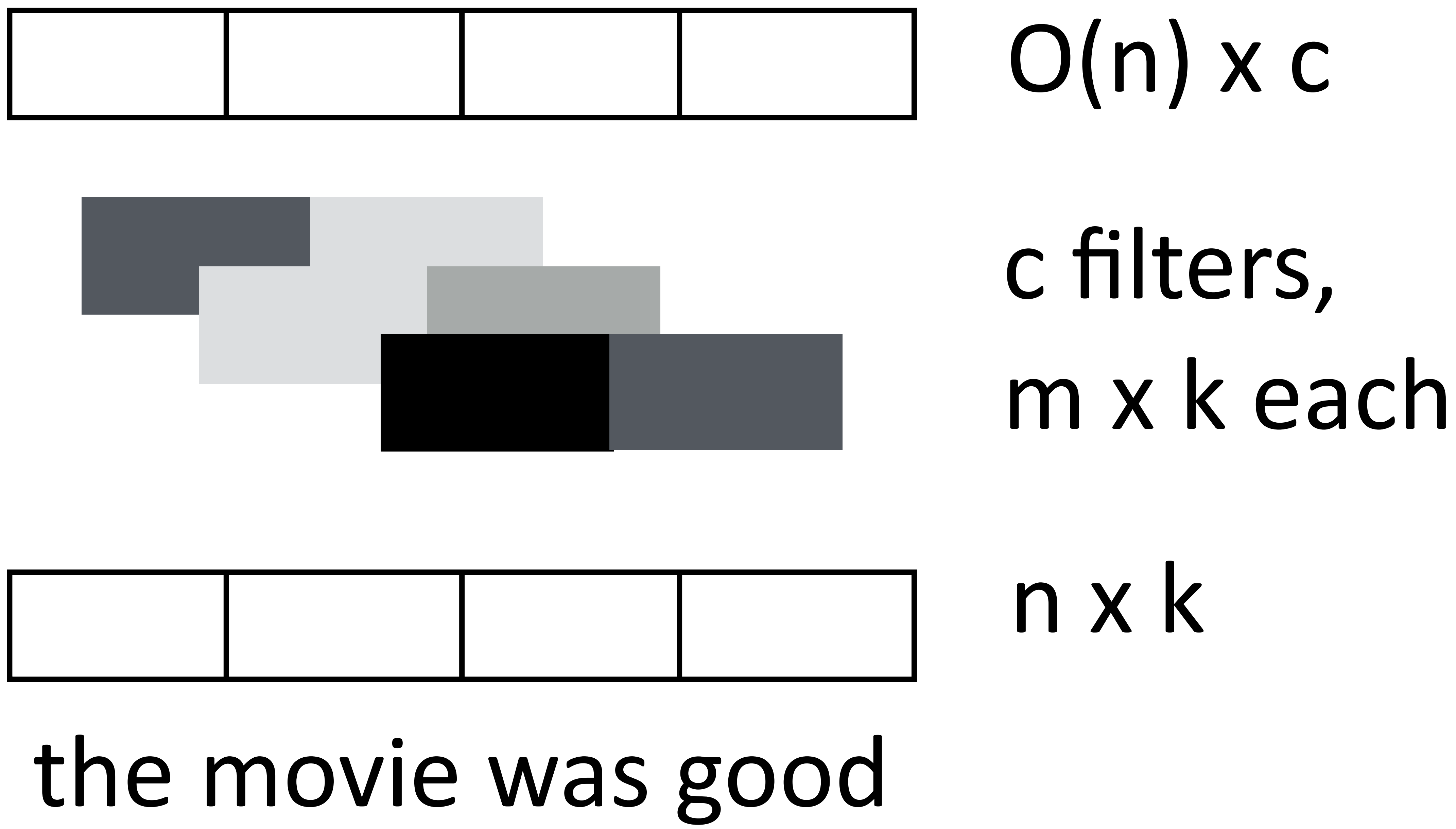
- Sequence-to-Sequence Model
- Attention Mechanism
- Neural MT & Other Applications
- Copy/Pointer Network
- Transformer Architecture

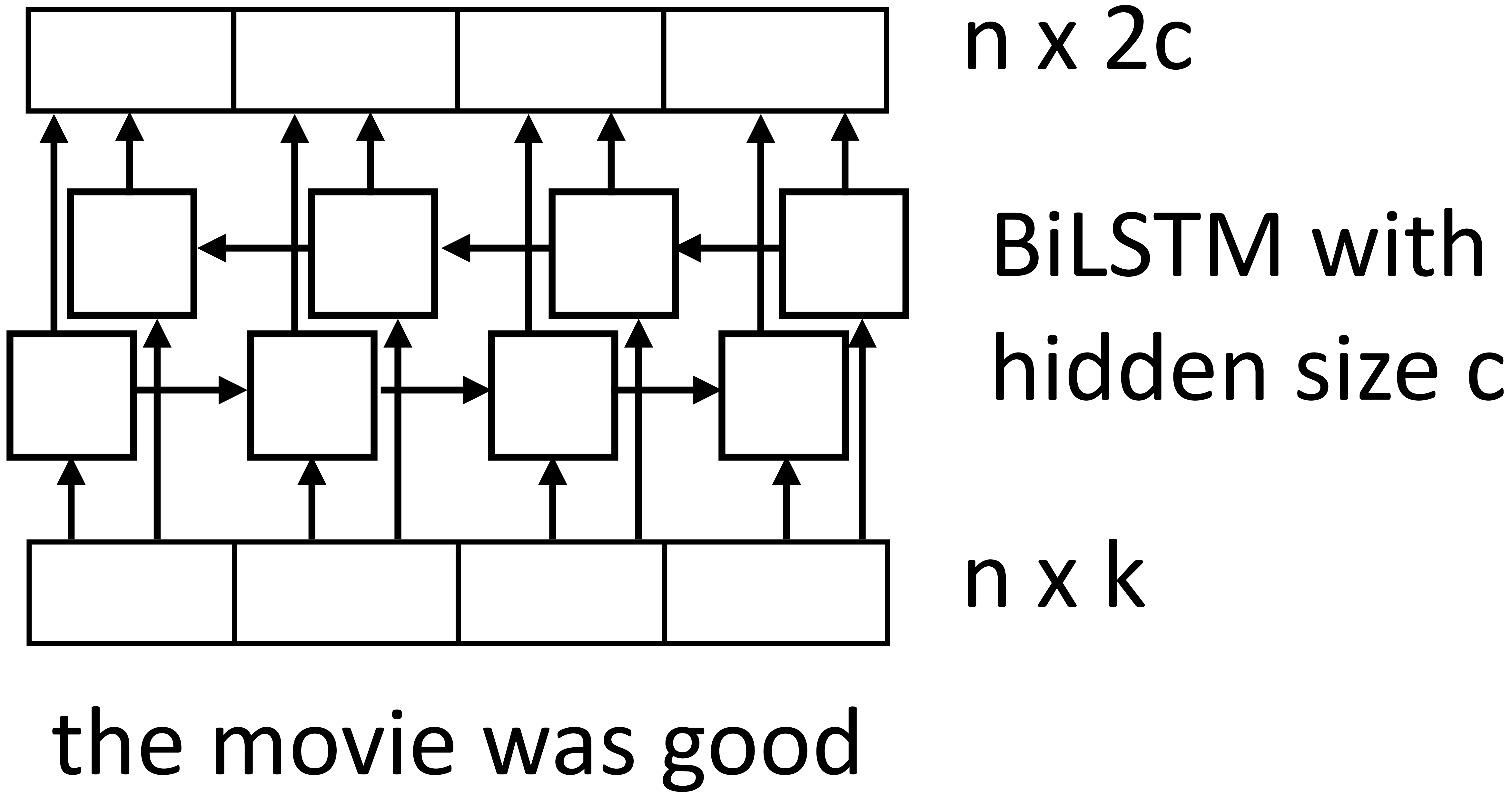
Administrivia

- Reading — Eisenstein 18.3-18.5
- Additional Reading — http://mt-class.org/jhu/



Recall: CNNs vs. LSTMs

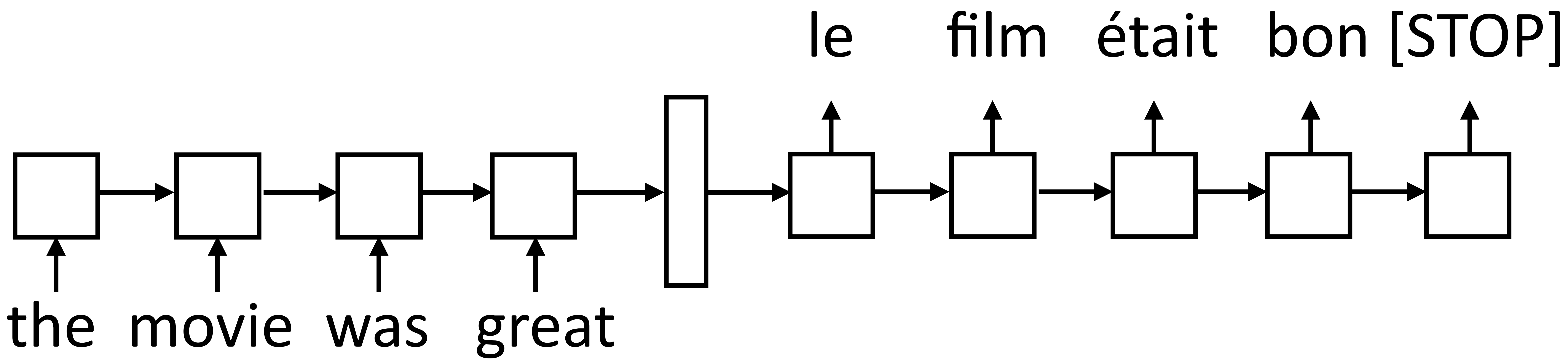




- Both LSTMs and convolutional layers transform the input using context
- LSTM: "globally" looks at the entire sentence (but local for many problems)
- CNN: local depending on filter width + number of layers

Encoder-Decoder

- Encode a sequence into a fixed-sized vector

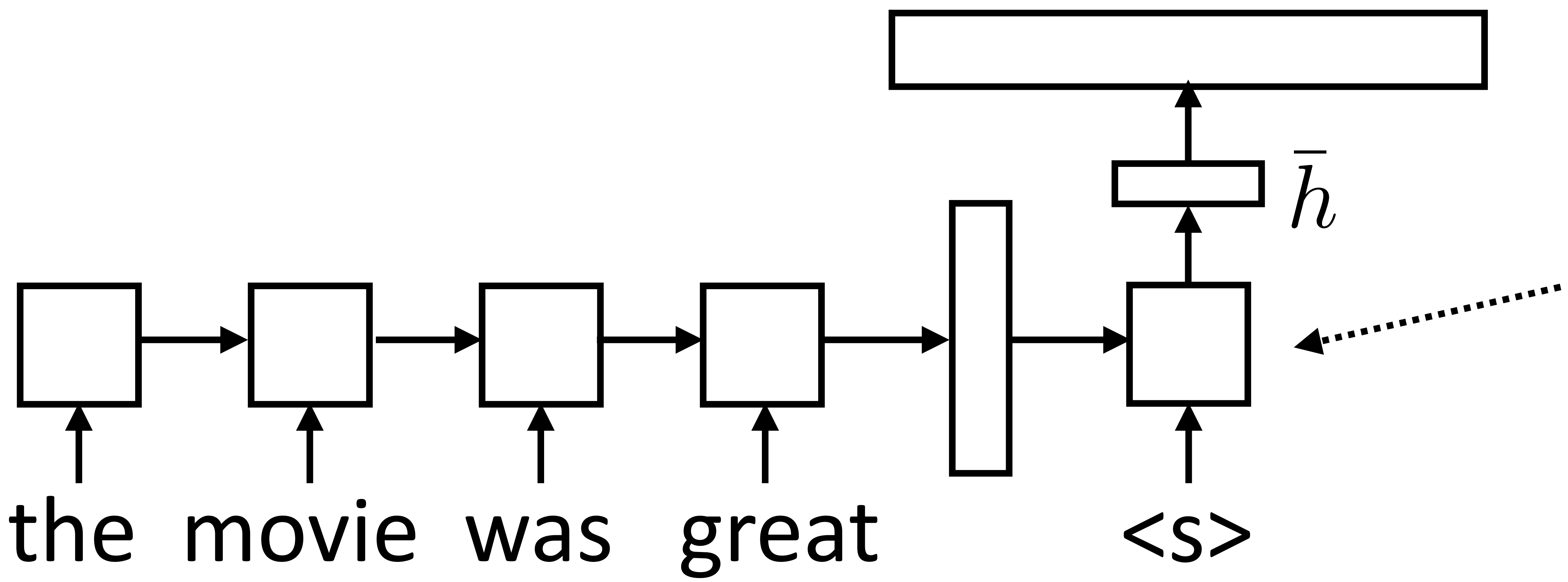


- Now use that vector to produce a series of tokens as output from a separate LSTM *decoder*
- Machine translation, NLG, summarization, dialog, and many other tasks (e.g., semantic parsing, syntactic parsing) can be done using this framework.

Sutskever et al. (2014)

Model

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



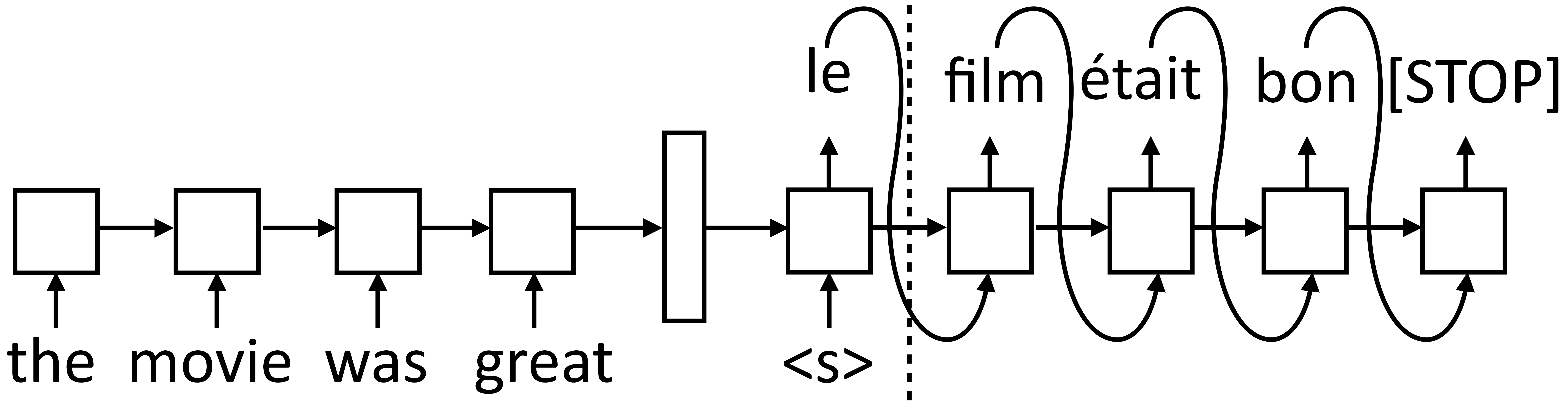
$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$

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

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

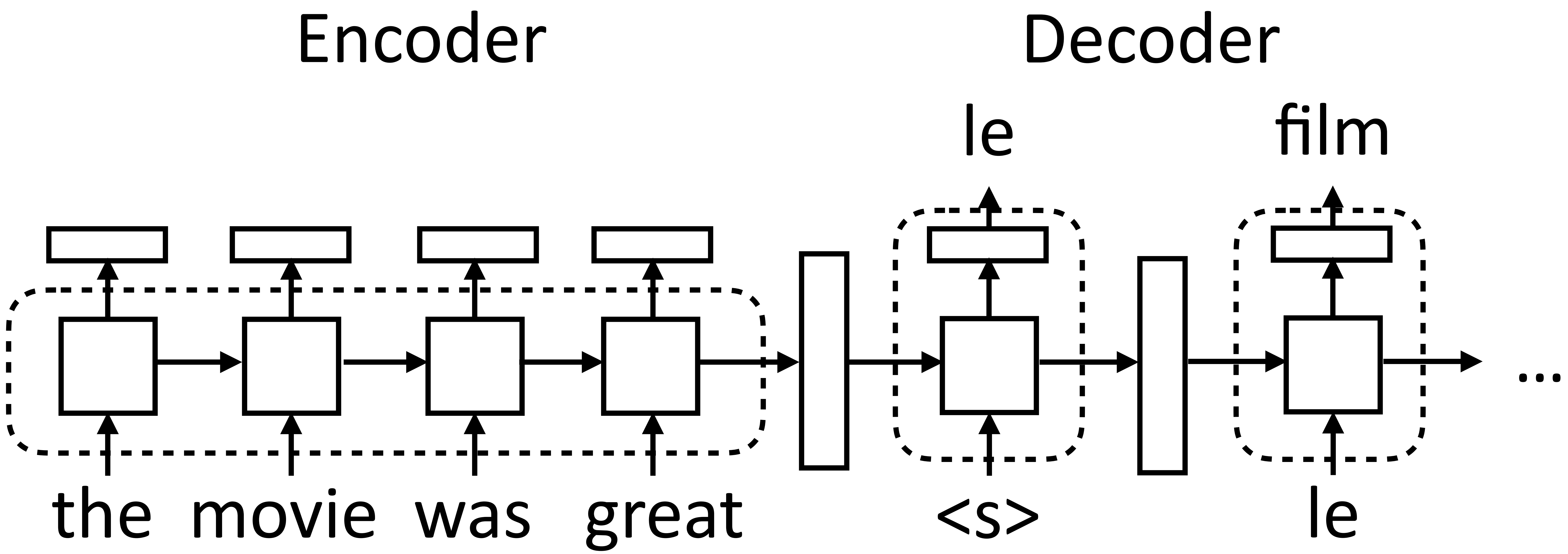
Inference

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



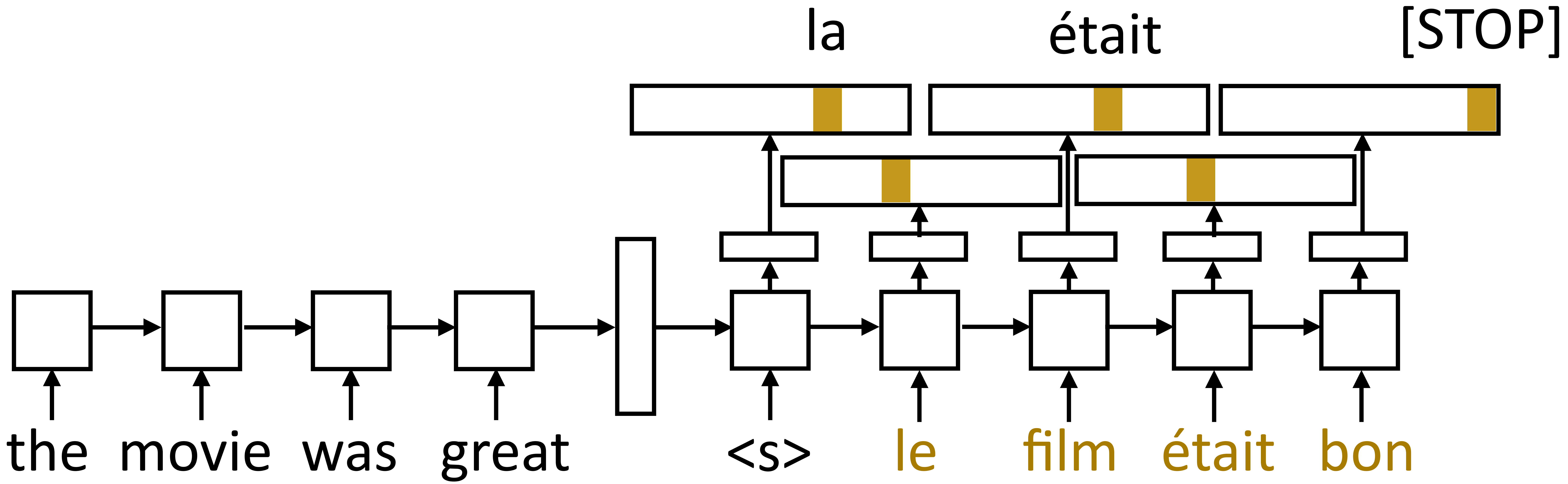
- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state
- Decoder is advanced one state at a time until [STOP] is reached

Implementing seq2seq Models



- 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

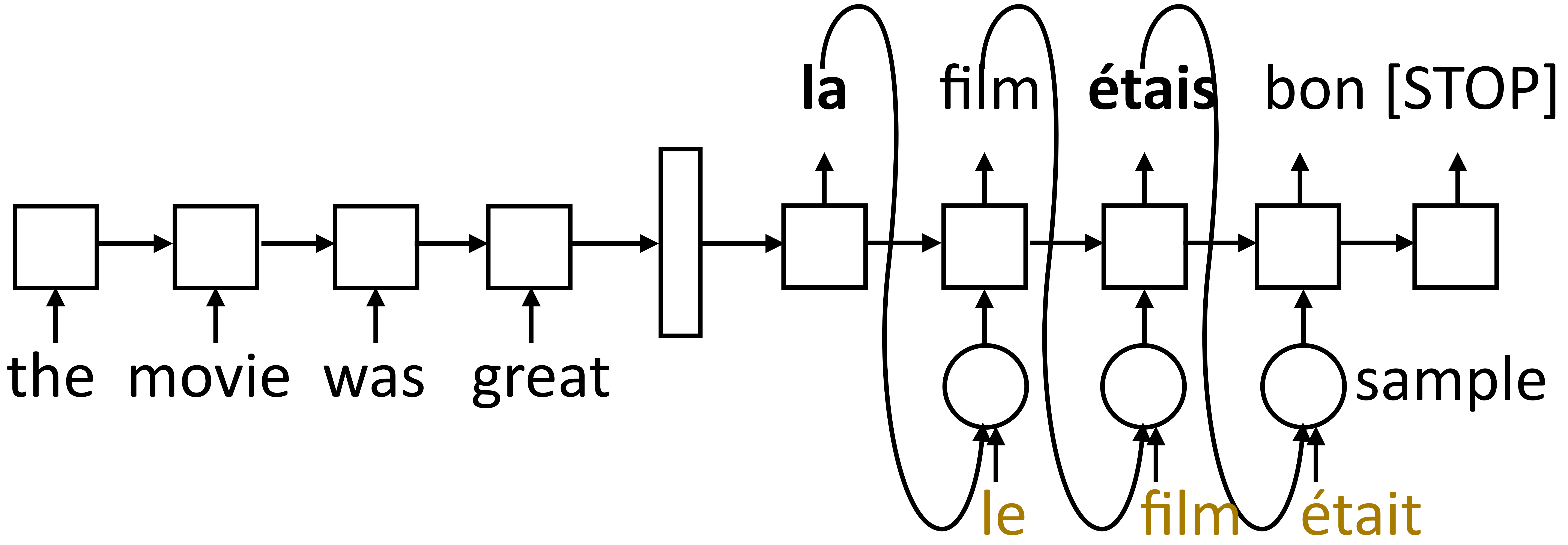
Training



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

Training: Scheduled Sampling

- Model needs to do the right thing even with its own predictions



- Scheduled sampling: with probability *p*, take the gold (human) translation as input, else take the model's prediction
- Starting with *p* = 1 and decaying it works best

Bengio et al. (2015)

Implementation Details

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

- Encoder: Can be a CNN/LSTM/Transformer…

- Batching is a bit tricky:
  - encoder should use pack_padded_sequence to handle different lengths.
  - The decoder should pad everything to the same length and use a mask to only accumulate “valid” loss terms

Implementation Details (cont')

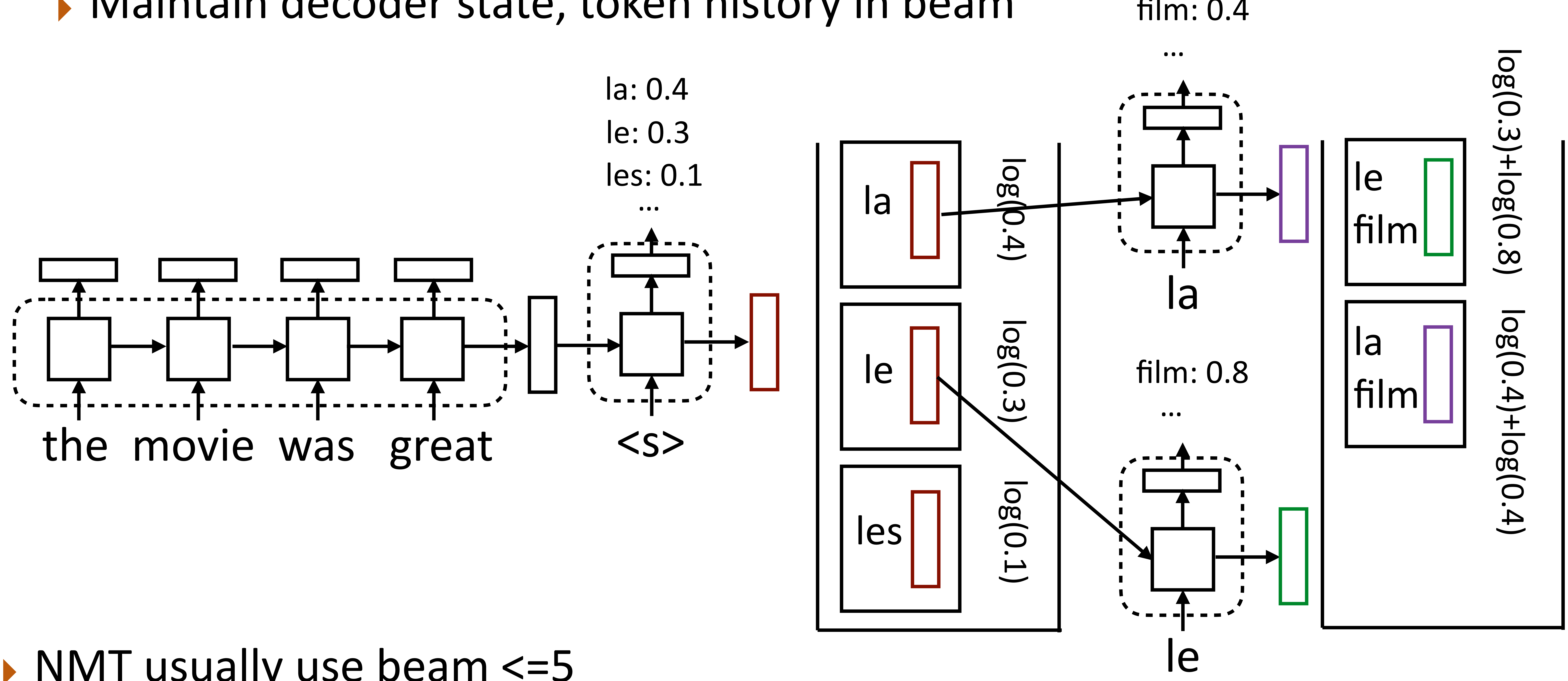
- Decoder:
  - Test time: execute one step of computation at a time, so computation graph is formulated as taking one input + hidden state. Until reach <STOP>.
  - Training time: you can execute all timesteps as part of one computation graph

- Beam search: can help with lookahead. Finds the (approximate) highest scoring sequence:

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

Beam Search

- Maintain decoder state, token history in beam



- NMT usually use beam <=5
- Keep **both** *film* states! Hidden state vectors are different

Meister et al. (2020)

Attention

Problems with Seq2seq Models

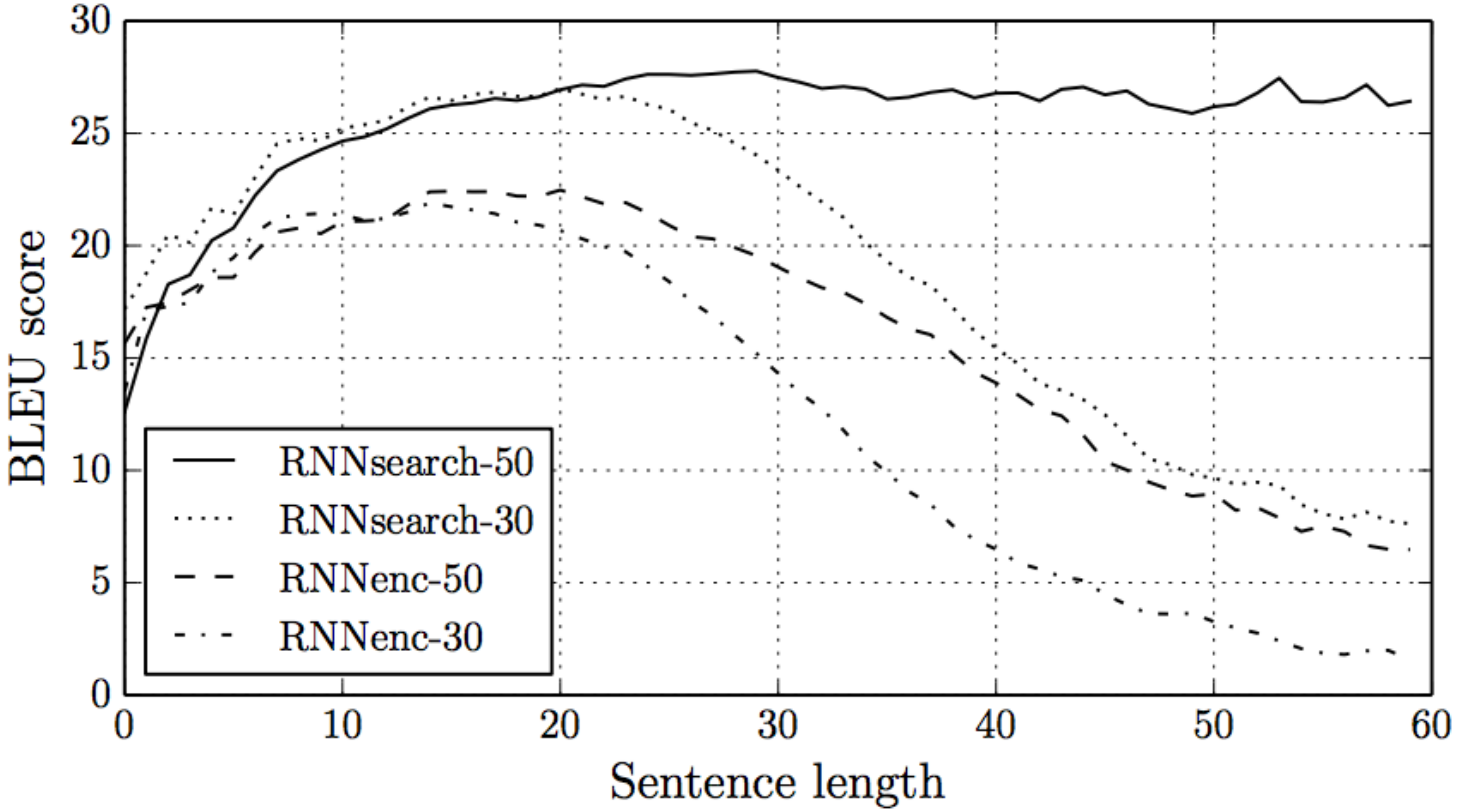
- Encoder-decoder models like to repeat themselves:

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

- Often a byproduct of training these models poorly. Input is forgotten by the LSTM so it gets stuck in a "loop" of generation the same output tokens again and again.
- 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 hidden representation doesn't scale; 2) LSTMs still have a hard time remembering for really long sentences



RNNenc: the model we've discussed so far

RNNsearch: uses attention

Bahdanau et al. (2014)

Problems with Seq2seq Models

- Unknown words:

en: The *ecotax* portico in *Pont-de-Buis* , . . . [truncated] . . . , was taken down on Thursday morning

fr: Le *portique* *écotaxe* de *Pont-de-Buis* , . . . [truncated] . . . , a été *démonté* jeudi matin

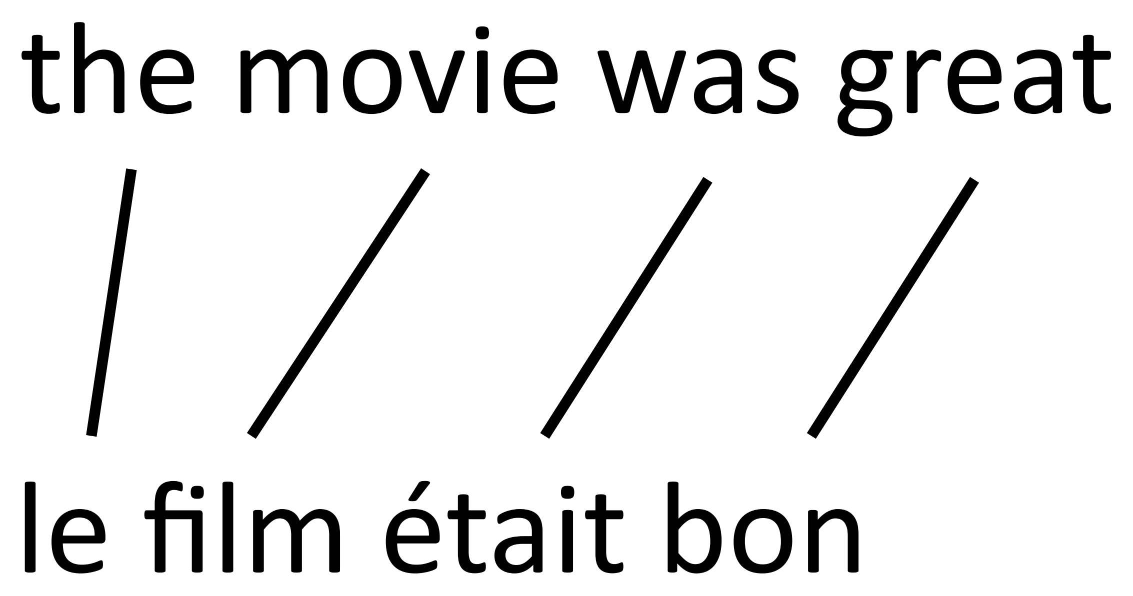
nn: Le unk de unk à unk , . . . [truncated] . . . , a été pris le jeudi matin

- Encoding these rare words into a vector space is really hard
- In fact, we don't want to encode them, we want a way of directly looking back at the input and copying them (Pont-de-Buis)

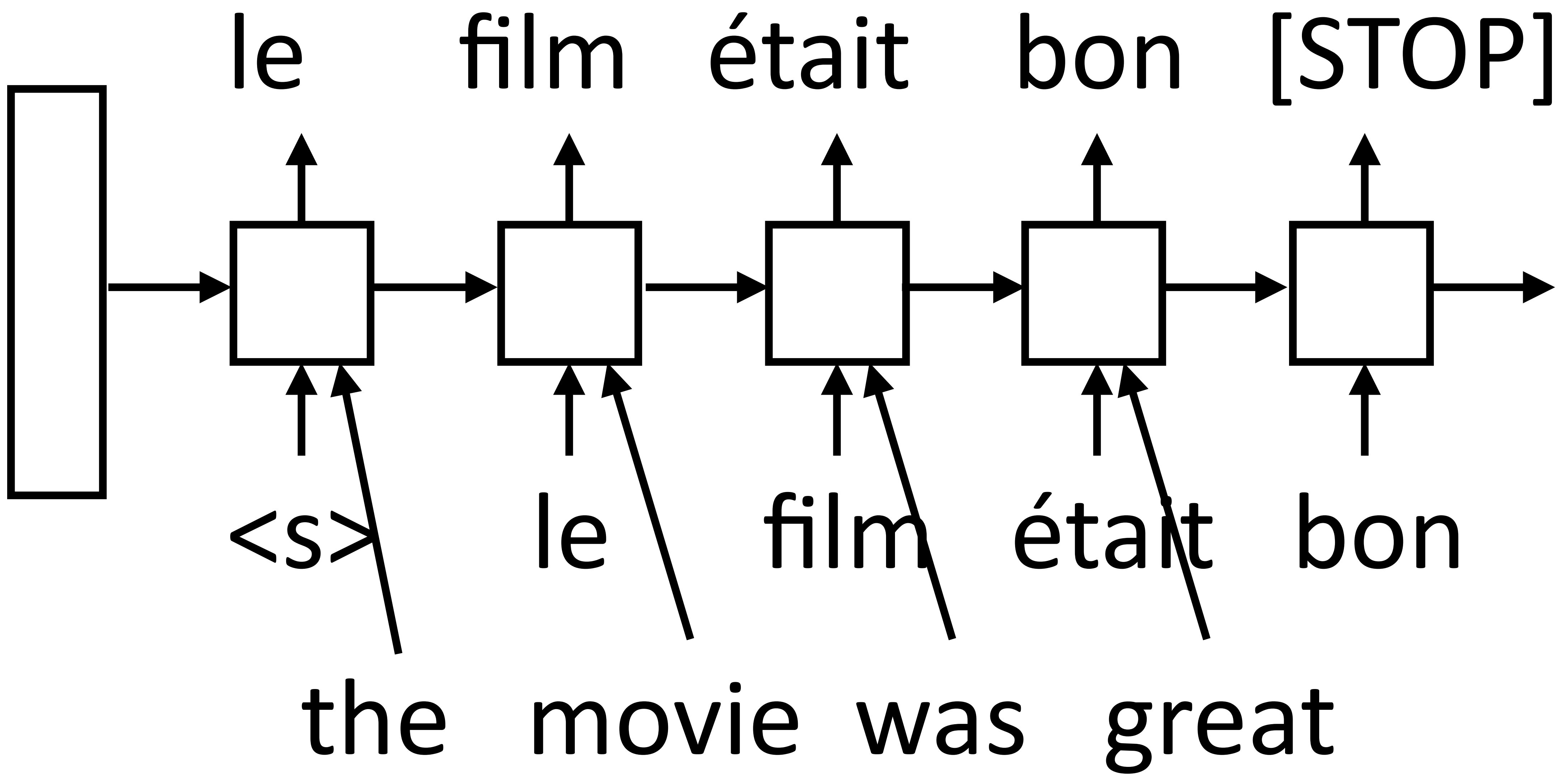
Jean et al. (2015), Luong et al. (2015)

Aligned Inputs

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

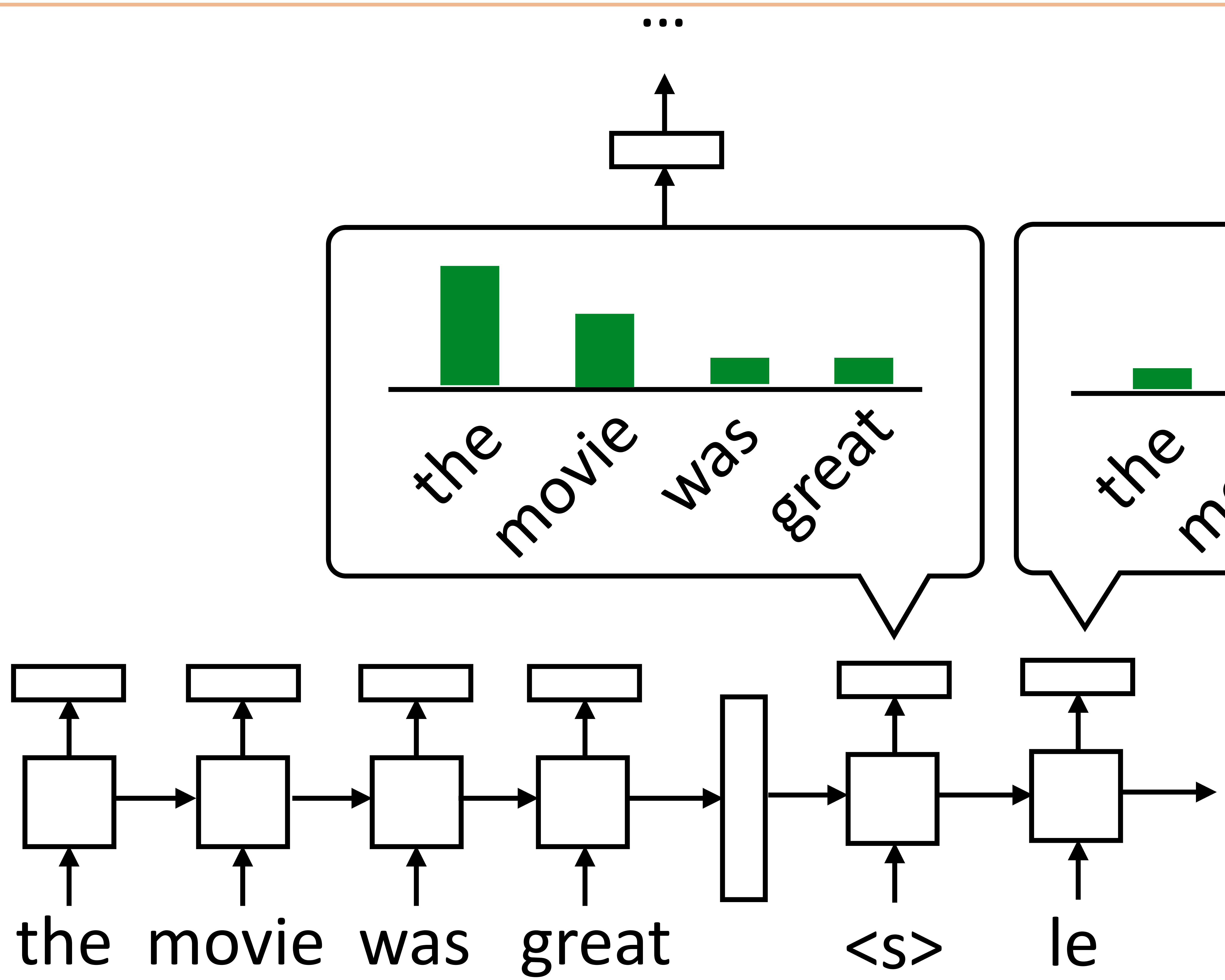


- In that case, we could look at the corresponding input word when translating — might improve handling of long sentences!



- How can we achieve this without hardcoding it?

Attention

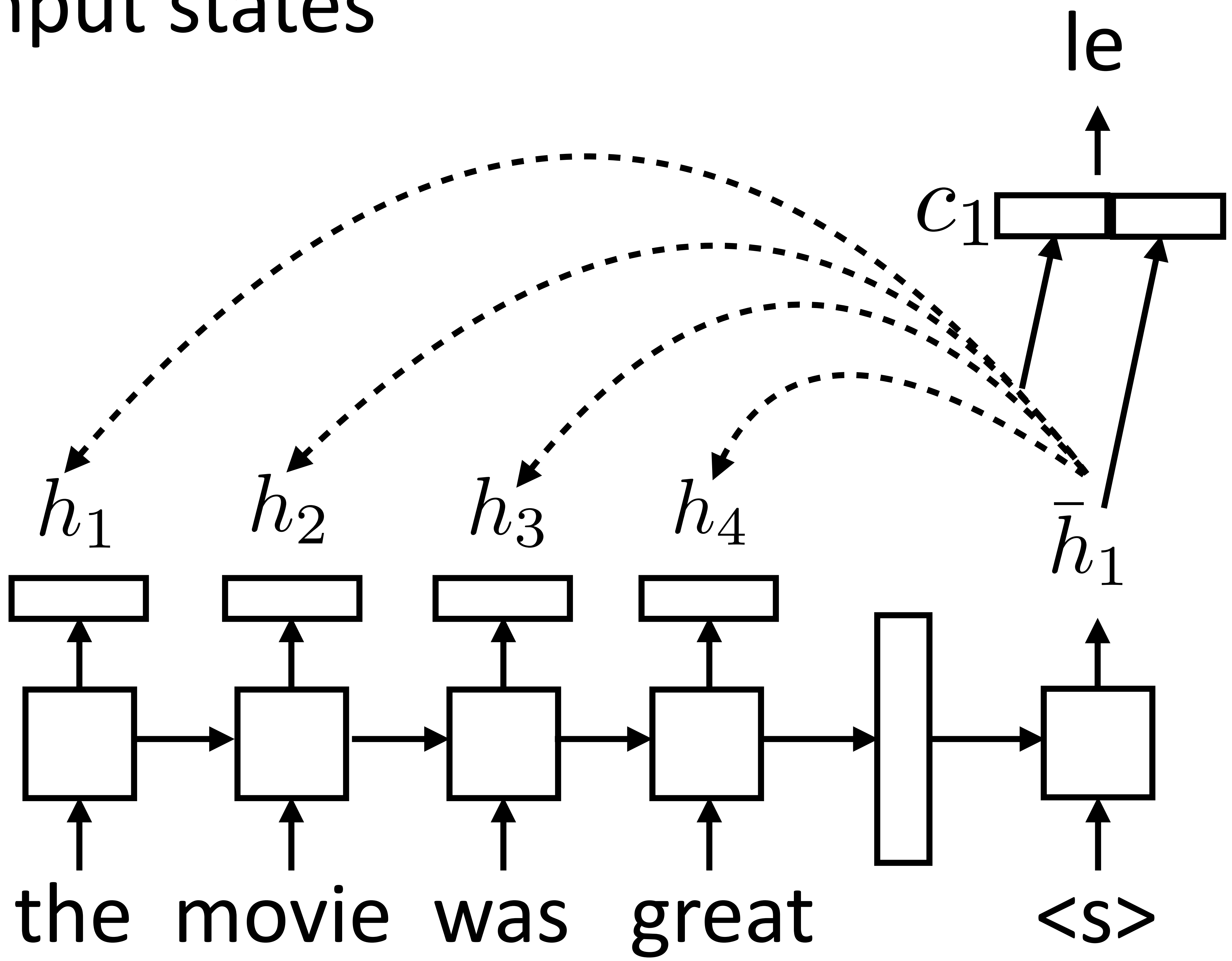


- At each decoder state, compute a distribution over source inputs based on current decoder state
- Use the weighted sum of input tokens to predict output

Attention

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

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

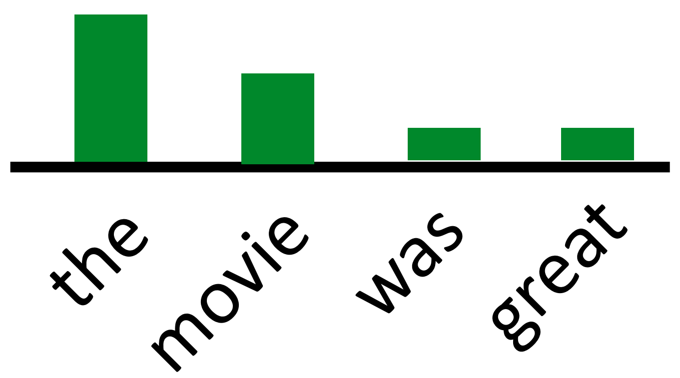


$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W[c_i; \bar{h}_i])$$

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

- Weighted sum of input hidden states (vector)

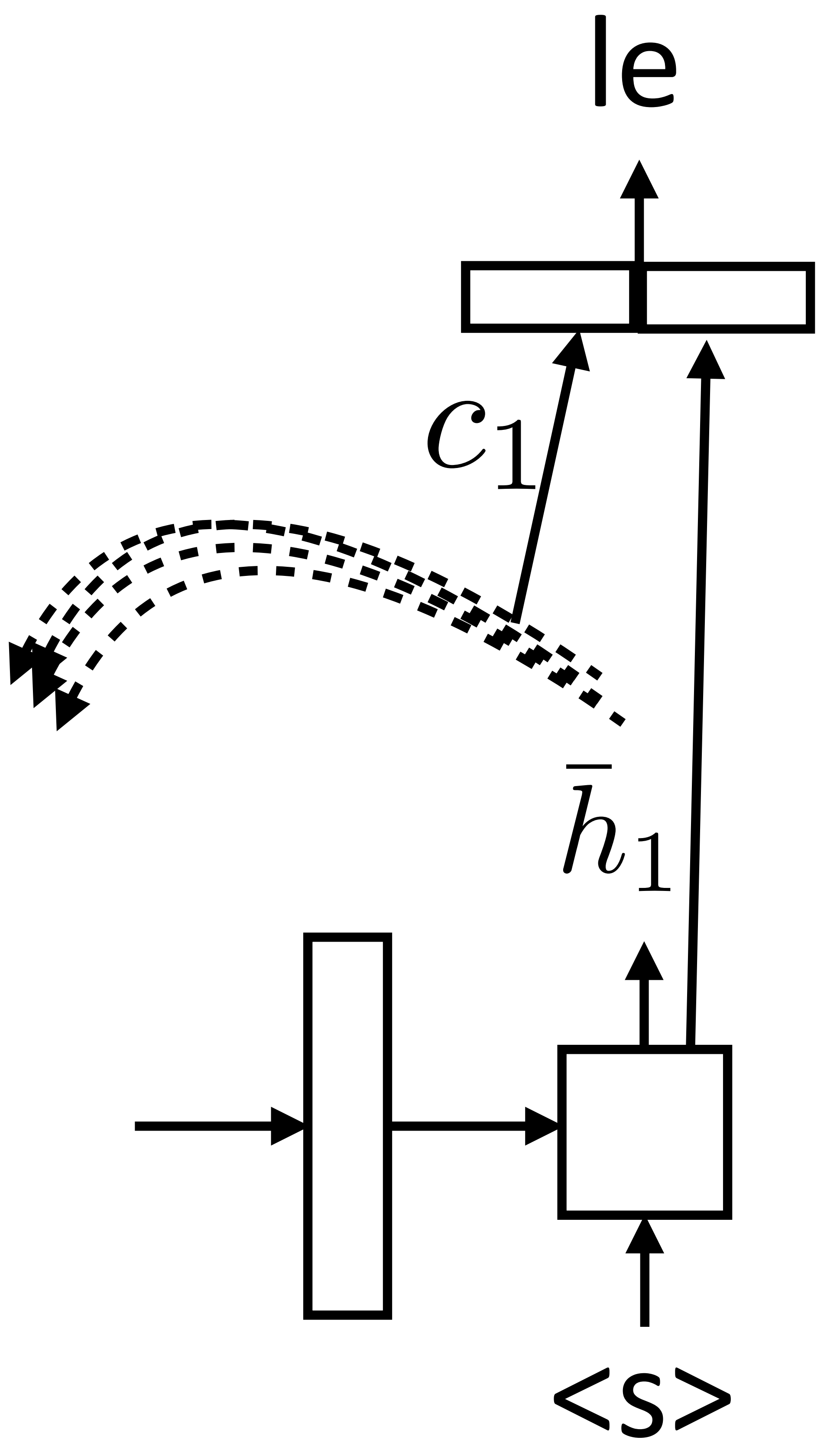
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$



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

- Some function f (next slide)

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

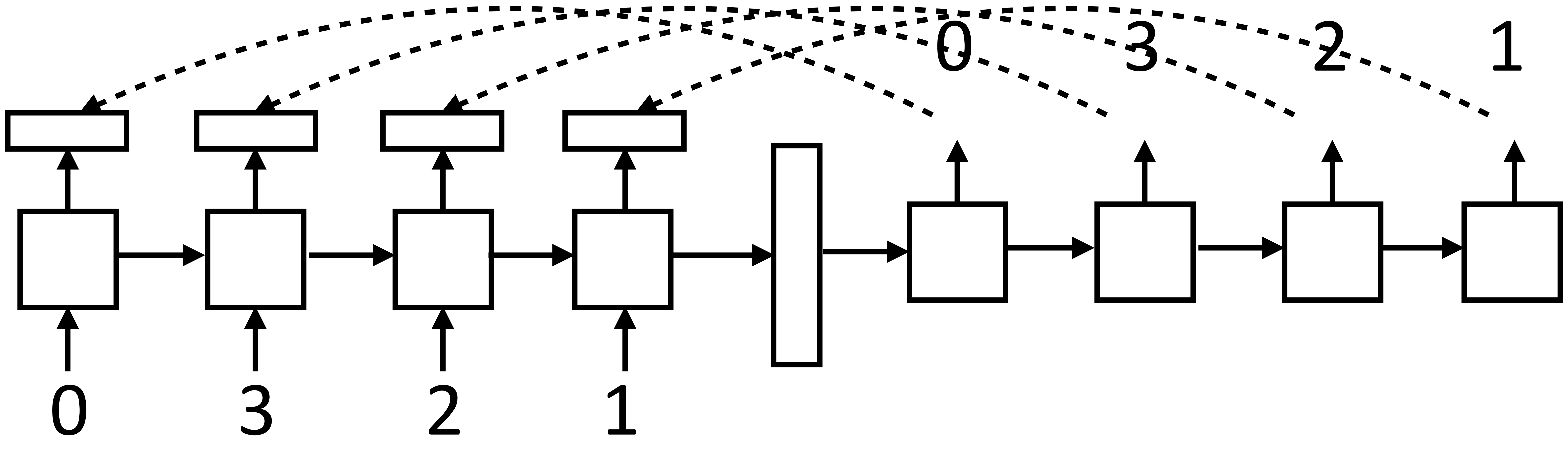
$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

- Luong+ (2015): bilinear

- Note that this all uses outputs of hidden layers

What can attention do?

- Learning to copy — how might this work?



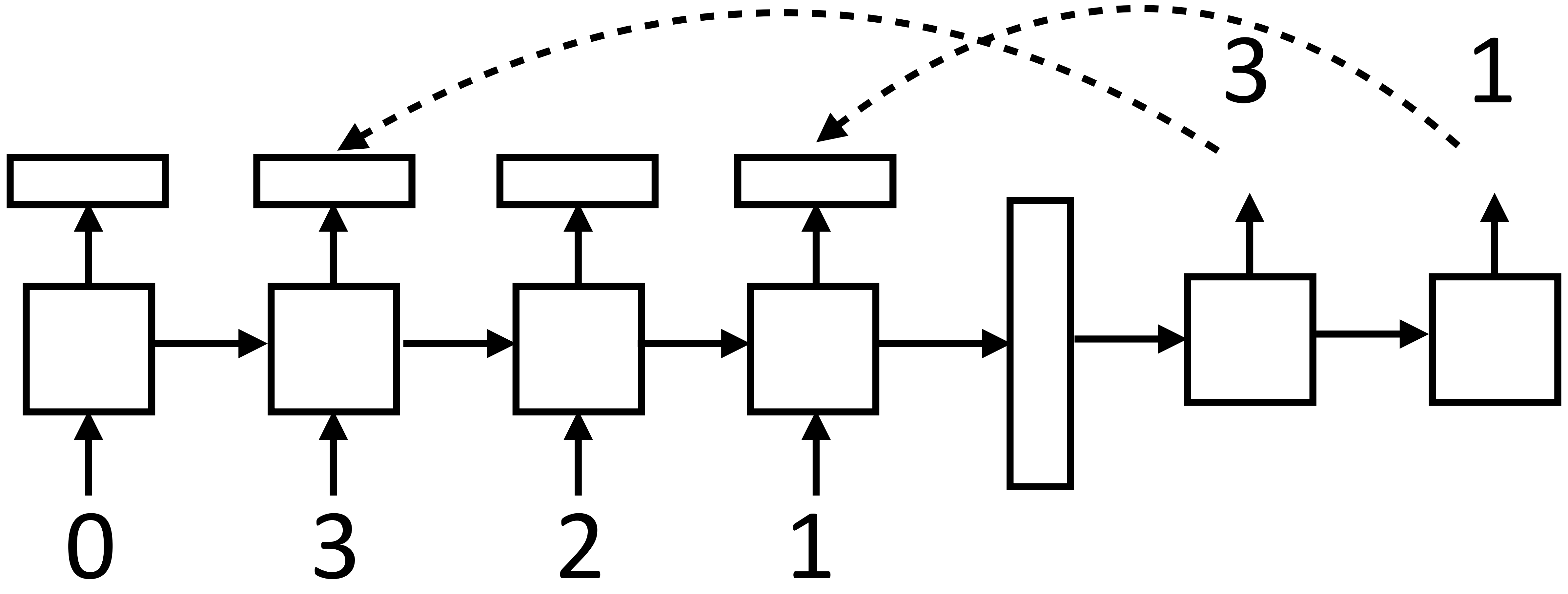
- LSTM can learn to count with the right weight matrix

- This is a kind of position-based addressing

Luong et al. (2015)

What can attention do?

- Learning to subsample tokens



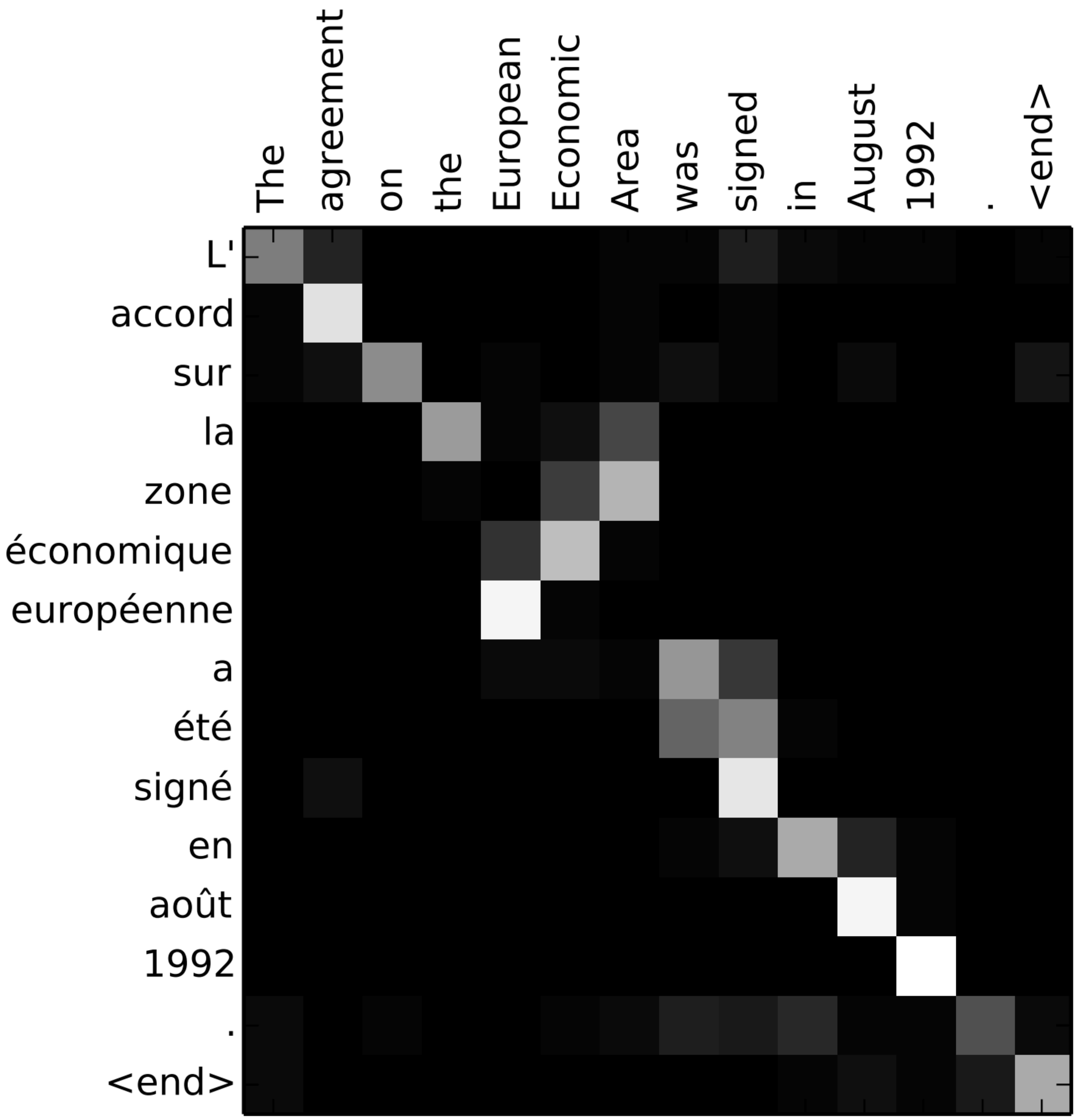
- Need to count (for ordering) and also determine which tokens are in/out

- Content-based addressing

Luong et al. (2015)

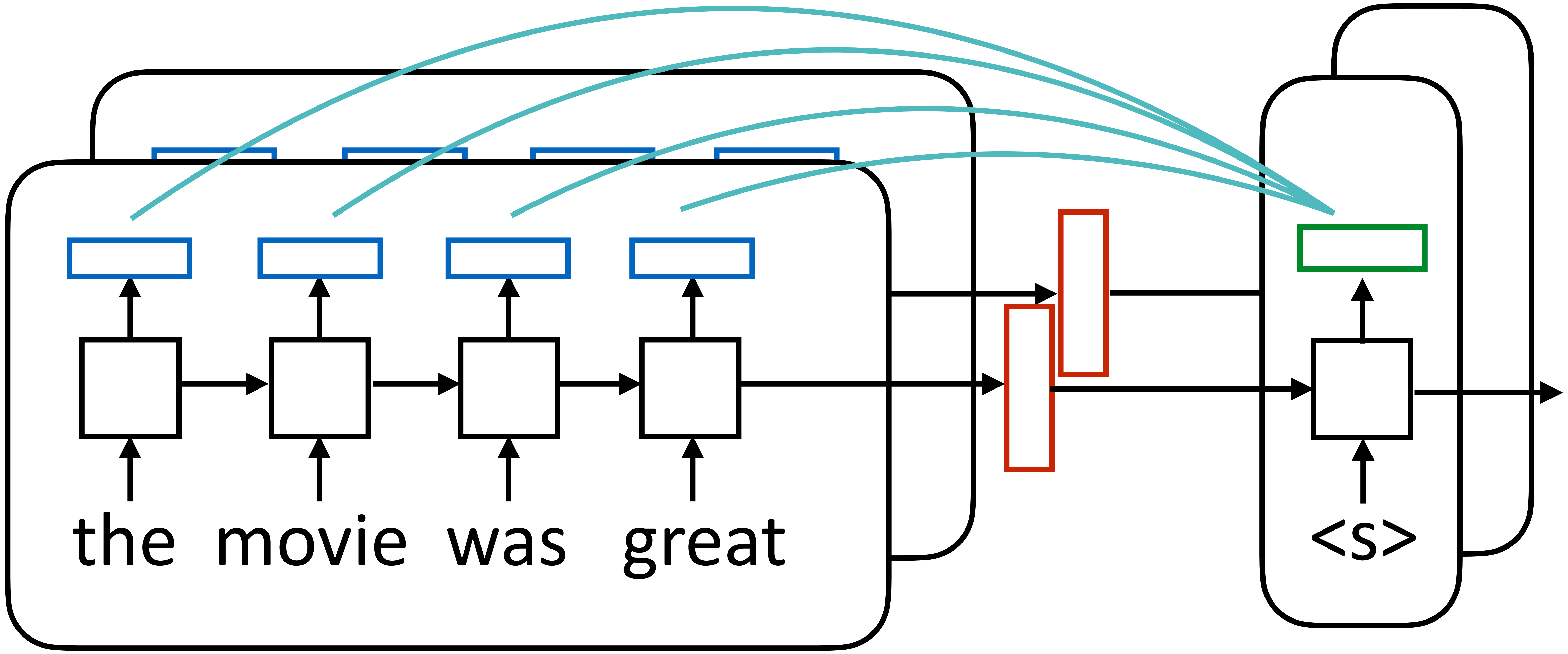
Attention

- Encoder hidden states capture contextual source word identity ("soft" word alignment)
- 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

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

- Machine translation: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with smarter attention (constrained to a small windows)

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

- Semantic parsing: ~30-50% accuracy -> 70+% accuracy on Geoquery

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

Neural MT

Encoder-Decoder MT

- Kalchbrenner & blunsom (2013), Bahanau et al. (2014), Cho et al. (2014)
- Sutskever et al. (2014) paper: first major application of LSTMs to NLP
- Basic encoder-decoder with beam search

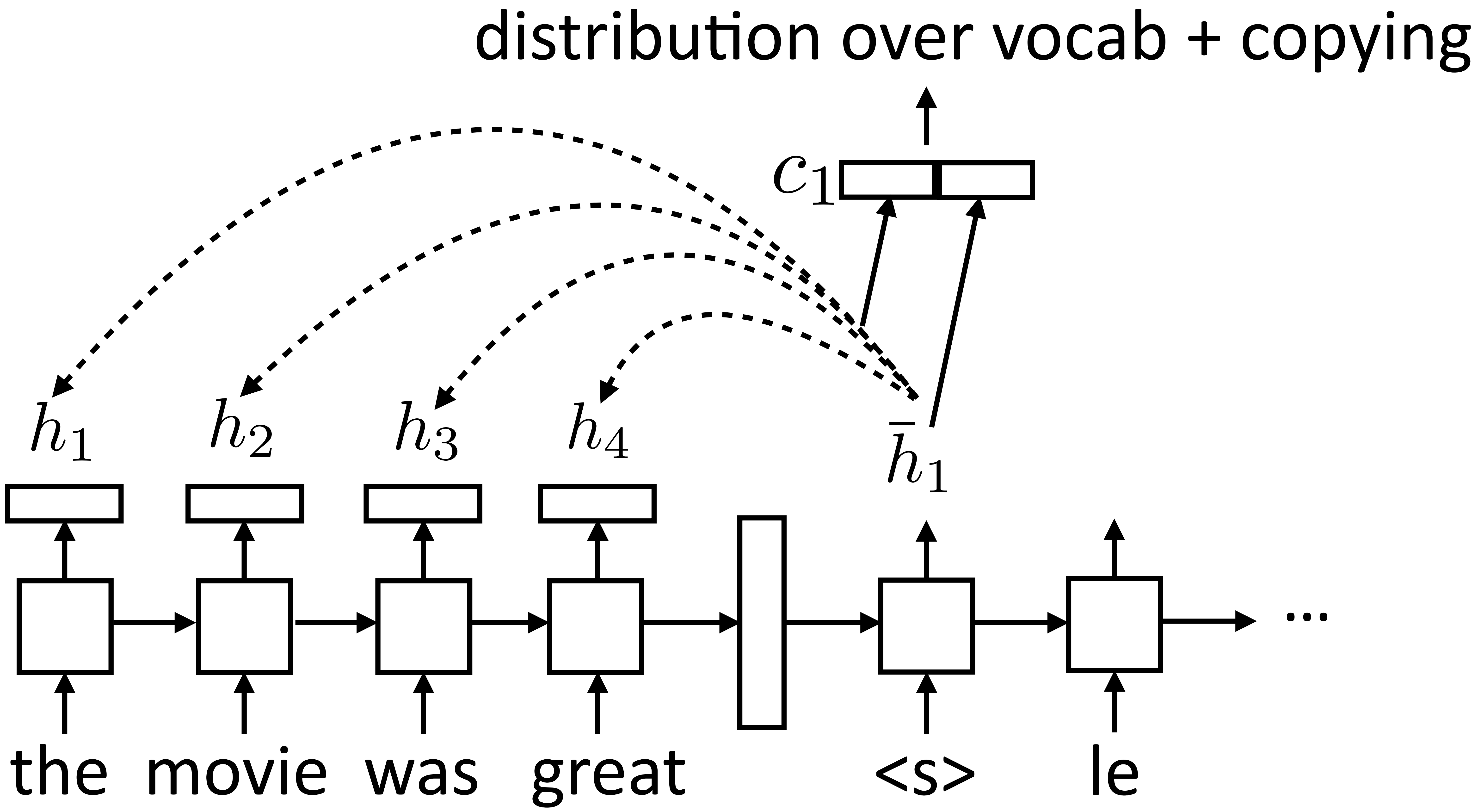
Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

- SOTA = 37.0 — not all that competitive…

Sutskever et al. (2014)

Encoder-Decoder MT

- Better encoder-decoder with attention and copying for rare words



Results: WMT English-French

- 12M sentence pairs

Classic phrase-based system: ~**33** BLEU, uses additional target-language data

Rerank with LSTMs: **36.5** BLEU (long line of work here; Devlin+ 2014)

Sutskever+ (2014) seq2seq single: **30.6** BLEU

Sutskever+ (2014) seq2seq ensemble: **34.8** BLEU

Luong+ (2015) seq2seq ensemble with attention and rare word handling: **37.5** BLEU

- But English-French is a really easy language pair and there's *tons* of data for it! Does this approach work for anything harder?

Results: WMT English-German

- 4.5M sentence pairs

Classic phrase-based system: **20.7** BLEU

Luong+ (2014) seq2seq: **14** BLEU

Luong+ (2015) seq2seq ensemble with rare word handling: **23.0** BLEU

- Not nearly as good in absolute BLEU, but not really comparable across languages
- French, Spanish = easiest
  German, Czech = harder
  Japanese, Russian = hard (grammatically different, lots of morphology…)

“Early” Neural MT

Effective Approaches to Attention-based Neural Machine Translation

Minh-Thang Luong **Hieu Pham** **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305
{lmthang,hyhieu,manning}@stanford.edu

Abstract

An attentional mechanism has lately been used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. However, there has been little work exploring useful architectures for attention-based NMT. This paper examines two simple and effective classes of attentional mechanism: a *global* approach which always attends to all source words and a *local* one that only looks at a subset of source words at a time. We demonstrate the effectiveness of both approaches on the

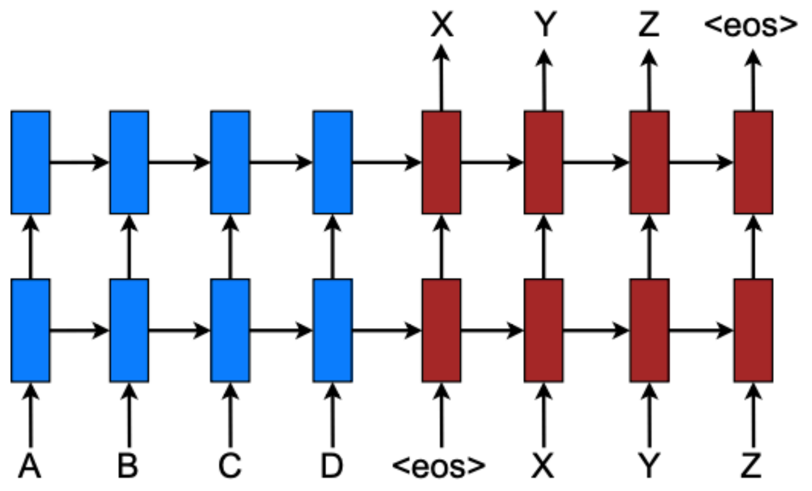


Figure 1: **Neural machine translation** – a stacking recurrent architecture for translating a source sequence `A B C D` into a target sequence `X Y Z`. Here, <`eos`> marks the end of a sentence.

ing plain SGD, (c) a simple learning rate schedule is employed – we start with a learning rate of 1; after 5 epochs, we begin to halve the learning rate every epoch, (d) our mini-batch size is 128, and (e) the normalized gradient is rescaled whenever its norm exceeds 5. Additionally, we also use dropout with probability 0.2 for our LSTMs as suggested by (Zaremba et al., 2015). For dropout models, we train for 12 epochs and start halving the learning rate after 8 epochs. For local attention models, we empirically set the window size $D = 10$.

Our code is implemented in MATLAB. When running on a single GPU device Tesla K40, we achieve a speed of 1K *target* words per second. It takes 7–10 days to completely train a model.

- TensorFlow first released in Nov 2015.
- PyTorch first released in 2016.

Luong et al. (2015)

MT Examples

src	In einem Interview sagte Bloom jedoch , dass er und Kerr sich noch immer lieben .
ref	However , in an interview , Bloom has said that he and *Kerr* still love each other .
best	In an interview , however , Bloom said that he and *Kerr* still love .
base	However , in an interview , Bloom said that he and **Tina** were still <unk> .

- best = with attention, base = no attention
- NMT systems can hallucinate words, especially when not using attention — phrase-based doesn't do this

Luong et al. (2015)

MT Examples

src	Wegen der von Berlin und der Europäischen Zentralbank verhängten strengen Sparpolitik in Verbindung mit der Zwangsjacke , in die die jeweilige nationale Wirtschaft durch das Festhalten an der gemeinsamen Währung genötigt wird , sind viele Menschen der Ansicht , das Projekt Europa sei zu weit gegangen
ref	The *austerity imposed by Berlin and the European Central Bank , coupled with the straitjacket* imposed on national economies through adherence to the common currency , has led many people to think Project Europe has gone too far .
best	Because of the strict *austerity measures imposed by Berlin and the European Central Bank in connection with the straitjacket* in which the respective national economy is forced to adhere to the common currency , many people believe that the European project has gone too far .
base	Because of the pressure **imposed by the European Central Bank and the Federal Central Bank with the strict austerity** imposed on the national economy in the face of the single currency , many people believe that the European project has gone too far .

- best = with attention, base = no attention

Luong et al. (2015)

Tokenization

Handling Rare Words

- Words are a difficult unit to work with: copying can be cumbersome, word vocabularies get very large

- Character-level models don’t work well

- Solution: “word pieces” (which may be full words but may be subwords)

Input: *_the* ***_eco tax*** *_port i co _in _Po nt - de - Bu is* …

Output: *_le _port ique* ***_éco taxe*** *_de _Pont - de - Bui s*

- Can help with transliteration; capture shared linguistic characteristics between languages (e.g., transliteration, shared word root, etc.)

Wu et al. (2016)

Byte Pair Encoding (BPE)

- Start with every individual byte (basically character) as its own symbol

```
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
```

- Count bigram character cooccurrences
- Merge the most frequent pair of adjacent characters

- Do this either over your vocabulary (original version) or over a large corpus (more common version)
- Final vocabulary size is often in 10k ~ 30k range for each language
- Most SOTA NMT systems use this on both source + target

Sennrich et al. (2016)

Word Pieces

- Alternative to BPE

while voc size < target voc size:

Build a language model over your corpus

Merge pieces that lead to highest improvement in language model perplexity

- SentencePiece library from Google: unigram LM
- Result: way of segmenting input appropriate for translation

Comparison

(a)	**Original:**	furiously		
	BPE:	_fur	iously	
	Unigram LM:	_fur	ious	ly

(b)	**Original:**	tricycles			
	BPE:	_t	ric	y	cles
	Unigram LM:	_tri	cycle	s	

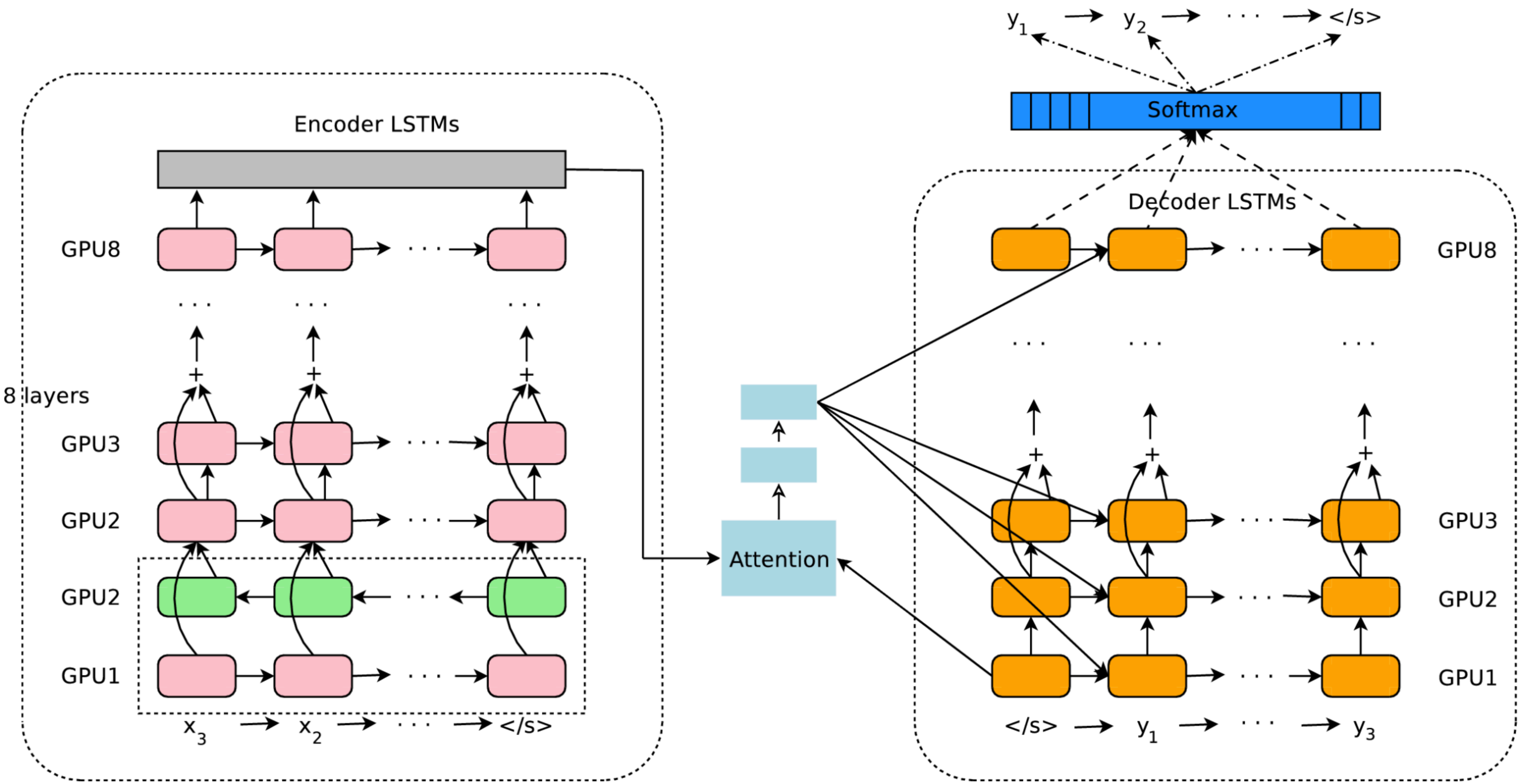
(c)	**Original:**	Completely preposterous suggestions									
	BPE:	_Comple	t	ely	_prep	ost	erous		_suggest	ions	
	Unigram LM:	_Complete	ly		_pre	post	er	ous	_suggestion	s	

- BPE produces less linguistically plausible units than word pieces (unigram LM)
- Some evidence that unigram LM works better in pre-trained transformer models

Bostrom and Durrett (2020)

Google NMT

Google's NMT System



- 8-layer LSTM encoder-decoder with attention, word piece vocabulary of 8k-32k

Wu et al. (2016)

Google’s NMT System

English-French:

Google’s phrase-based system: 37.0 BLEU

Luong+ (2015) seq2seq ensemble with rare word handling: 37.5 BLEU

Google’s 32k word pieces: 38.95 BLEU

English-German:

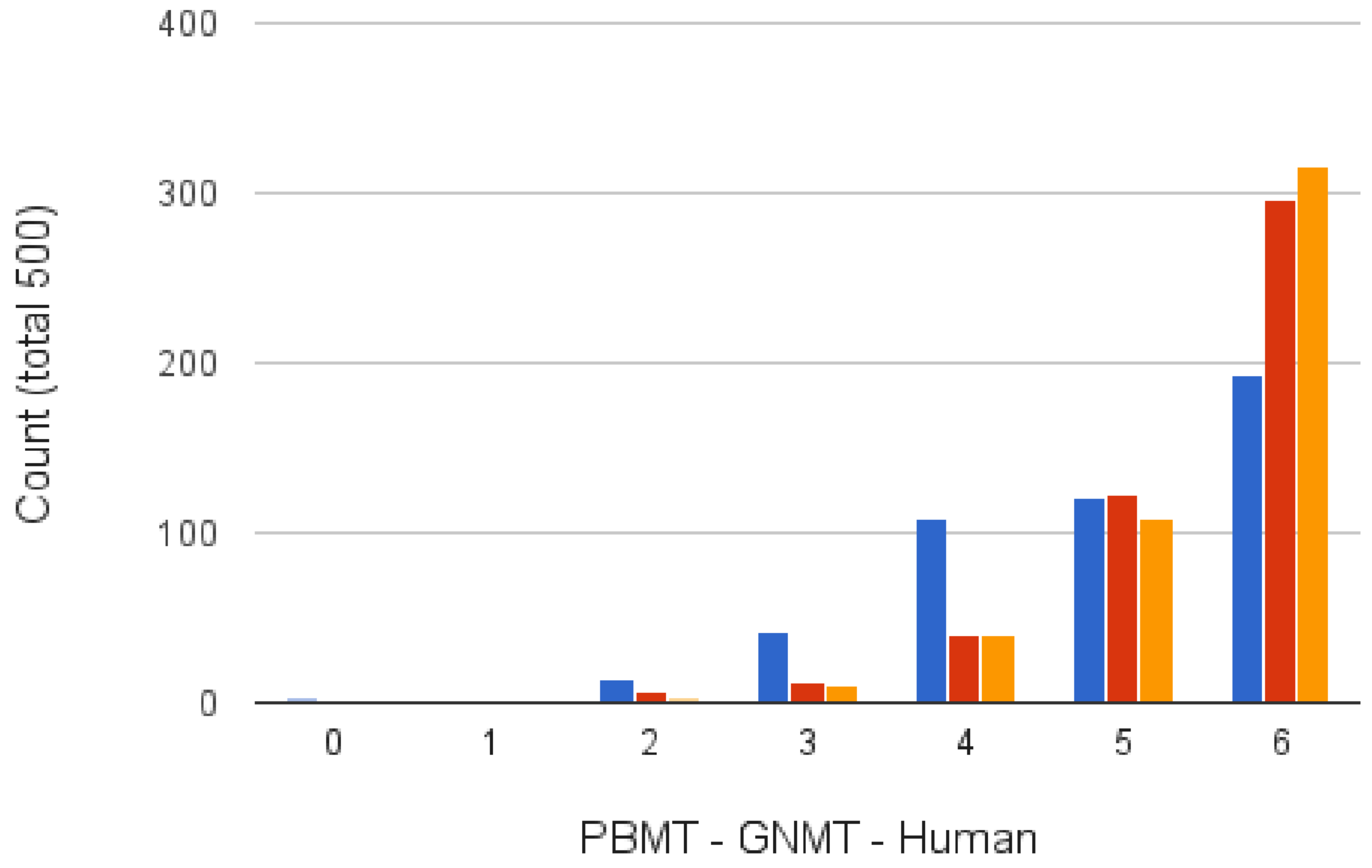
Google’s phrase-based system: 20.7 BLEU

Luong+ (2015) seq2seq ensemble with rare word handling: 23.0 BLEU

Google’s 32k word pieces: 24.2 BLEU

Wu et al. (2016)

Human Evaluation (En-Es)



- Similar to human-level performance *on English-Spanish*

Figure 6: Histogram of side-by-side scores on 500 sampled sentences from Wikipedia and news websites for a typical language pair, here English → Spanish (PBMT blue, GNMT red, Human orange). It can be seen that there is a wide distribution in scores, even for the human translation when rated by other humans, which shows how ambiguous the task is. It is clear that GNMT is much more accurate than PBMT.

Wu et al. (2016)

Google's NMT System

Source	She was spotted three days later by a dog walker trapped in the quarry	
PBMT	Elle a été repéré trois jours plus tard par un promeneur de chien piégé dans la carrière	6.0
GNMT	Elle a été repérée trois jours plus tard par un traîneau à chiens piégé dans la carrière.	2.0
Human	Elle a été repérée trois jours plus tard par une personne qui promenait son chien coincée dans la carrière	5.0

Gender is correct in GNMT but not in PBMT

"sled"

"walker"

The right-most column shows the human ratings on a scale of 0 (complete nonsense) to 6 (perfect translation)

Wu et al. (2016)

Frontiers in MT

Low-Resource MT

- Particular interest in deploying MT systems for languages with little or no parallel data

Burmese, Indonesian, Turkish

- BPE allows us to transfer models even without training on a specific language

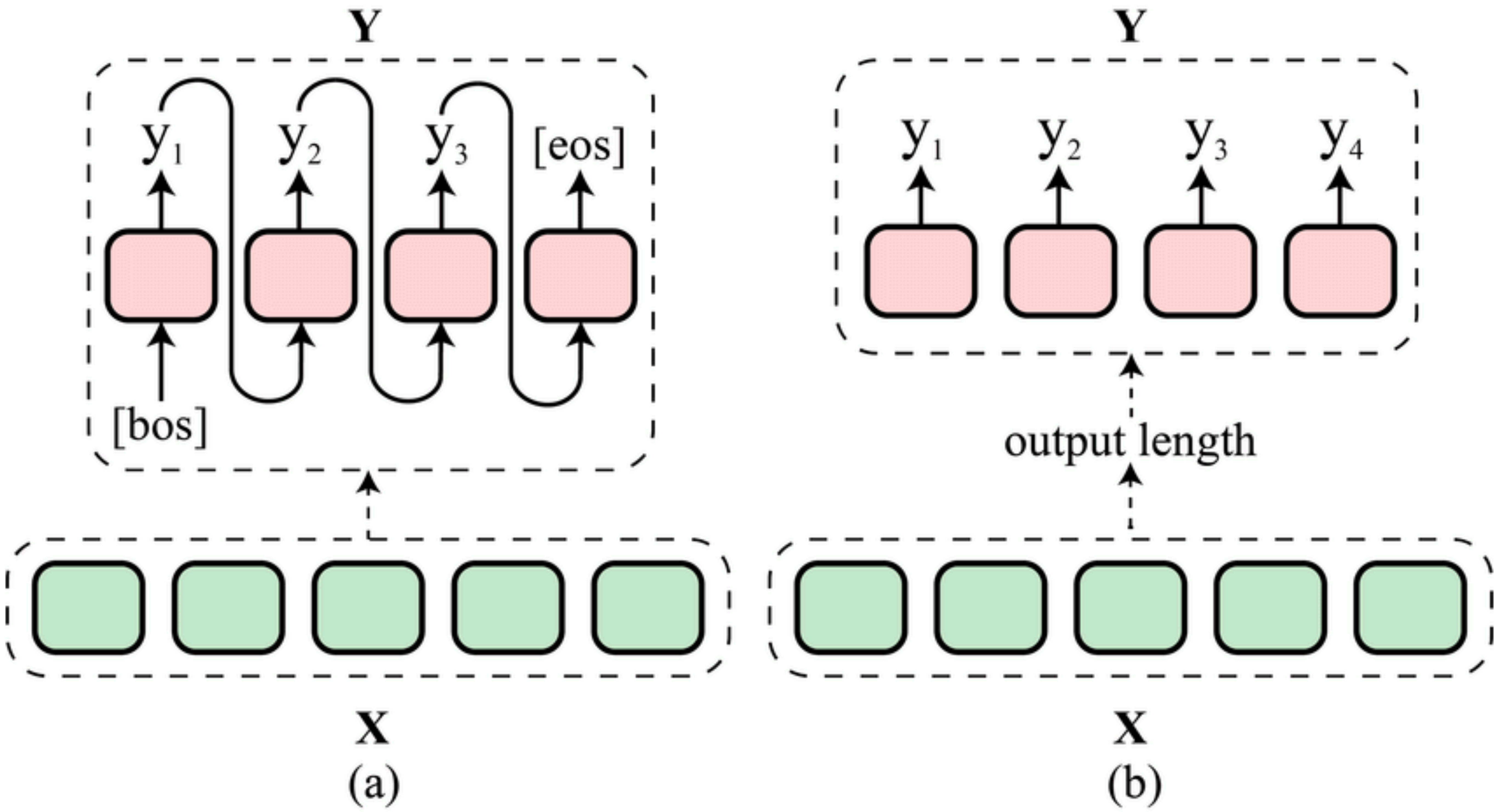
- Pre-trained models can help further

Transfer	BLEU My→En	Id→En	Tr→En
baseline (no transfer)	4.0	20.6	19.0
transfer, train	17.8	27.4	20.3
transfer, train, reset emb, train	13.3	25.0	20.0
transfer, train, reset inner, train	3.6	18.0	19.1

Table 3: Investigating the model's capability to restore its quality if we reset the parameters. We use En→De as the parent.

Aji et al. (2020)

Non-Autoregressive NMT



- Q: why non-autoregressive? Pros and cons?

Gu et al. (2018), Ghazvininejad et al. (2019), Kasai et al. (2020)

Unsupervised MT

Approach	Train/Val	Test	Loss
Supervised MT	L1-L2	L1-L2	$\mathcal{L}^{MT}_{x \to y} = \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim (\mathcal{X},\mathcal{Y})} \left[-\log p_{x \to y}(\mathbf{y}\|\mathbf{x}) \right]$
Unsupervised MT	L1, L2	L1-L2	$\mathcal{L}^{BT}_{x \leftrightarrow y} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[-\log p_{y \to x}(\mathbf{x}\|g^*(\mathbf{x})) \right] + \mathbb{E}_{\mathbf{y} \sim \mathcal{Y}} \left[-\log p_{x \to y}(\mathbf{y}\|h^*(\mathbf{y})) \right]$ g^*, h^*: sentence predictors

- Common principles of unsupervised MT
  - Language models
  - (Iterative) Back-translation!

Lample et al. (2018)

Takeaways

- Can build MT systems with LSTM encoder-decoders, CNNs, or Transformers

- Word piece / byte pair models are really effective and easy to use

- State of the art systems are getting pretty good, but lots of challenges remain, especially for low-resource settings