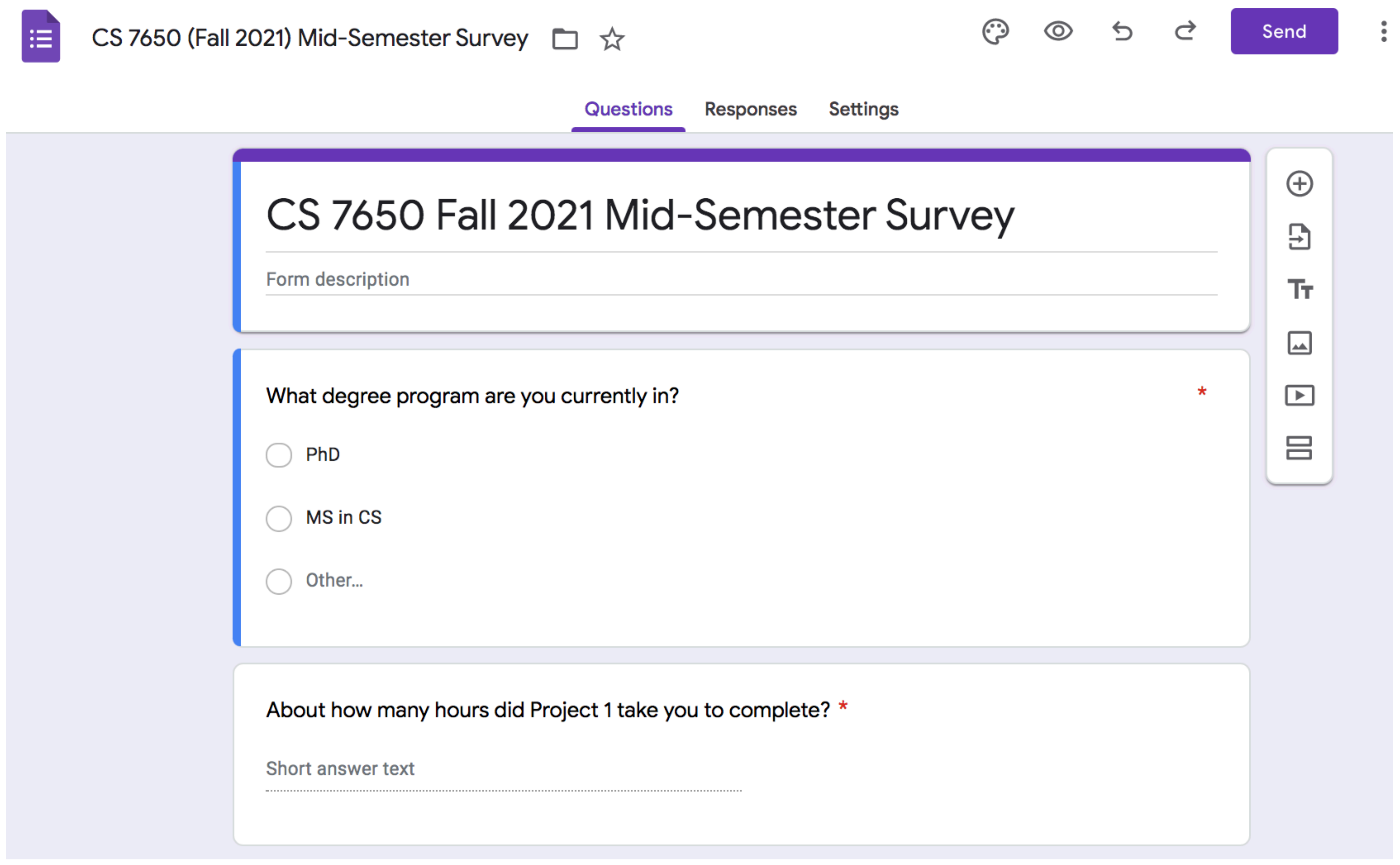
Neural MT + Copy/Pointer

Wei Xu

(many slides from Greg Durrett)

Administrivia

- Midterm will be released soon (take-home)
- Mid-semester Survey (by Monday 10/25)



CS 7650 Fall 2021 Mid-Semester Survey

Form description

What degree program are you currently in? *

- PhD
- MS in CS
- Other...

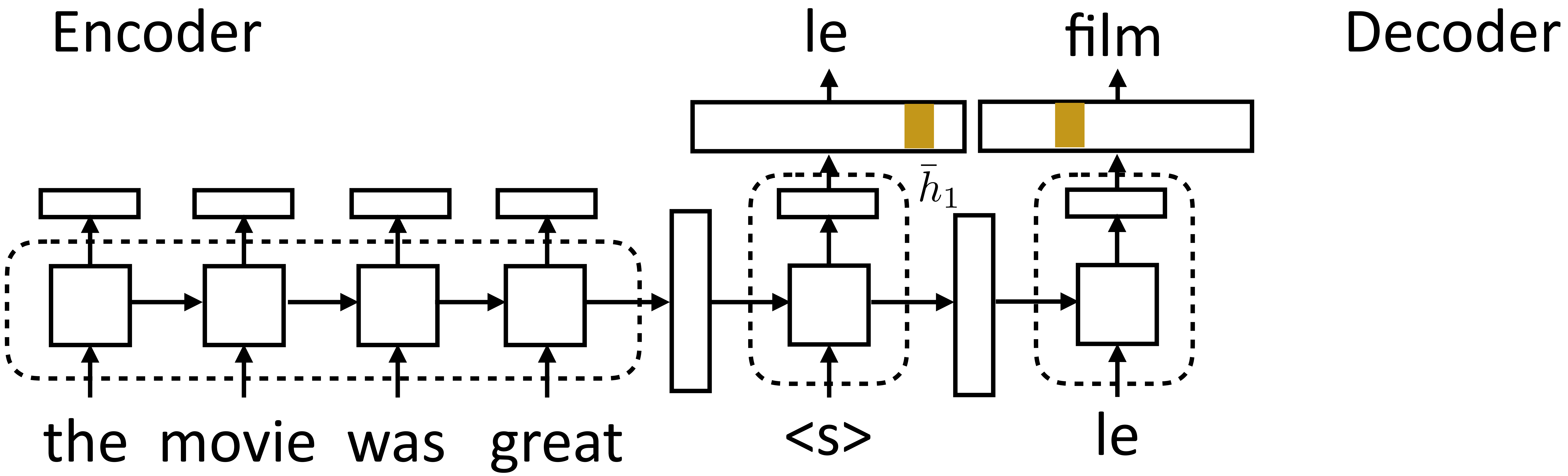
About how many hours did Project 1 take you to complete? *

Short answer text

This Lecture

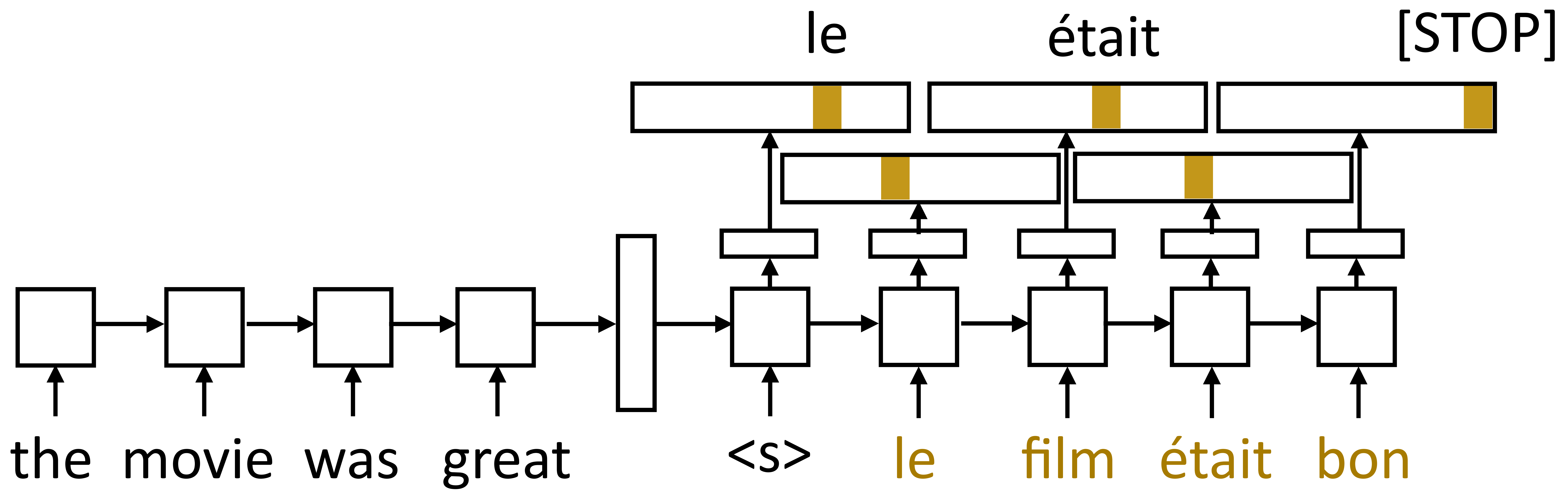
- Sequence-to-Sequence Model
- Attention Mechanism
- **Neural MT and other Applications**
- **Copy/Pointer Network**
- **Transformer Architecture (if time)**

Recap: Seq2Seq Model



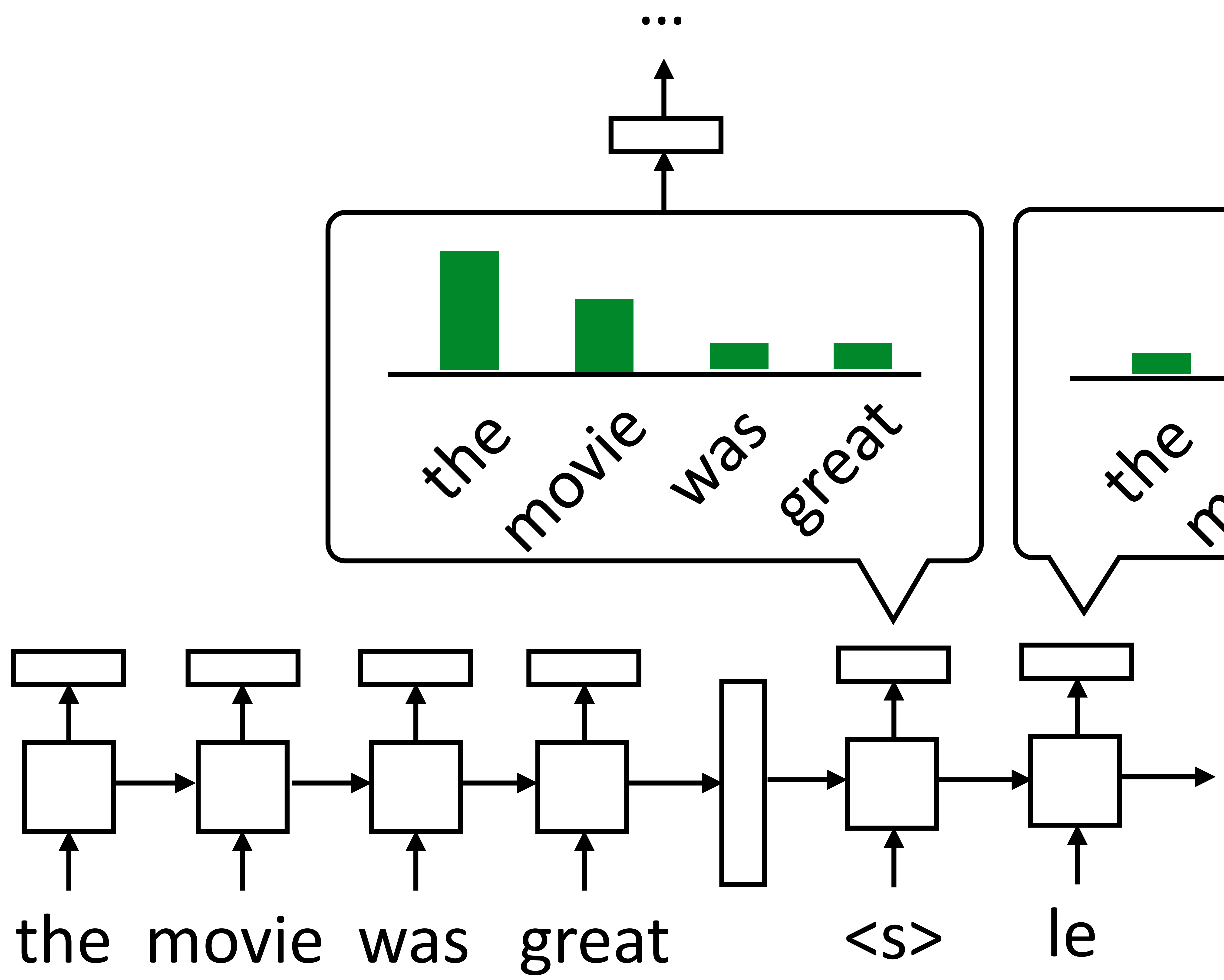
- Encoder: consumes sequence of tokens, produces a vector. Analogous to encoders for classification/tagging tasks $P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W\bar{h}_i)$
- Decoder: separate module, single cell. Takes two inputs: hidden state (vector *h* or tuple (*h*, *c*)) and previous token. Outputs token + new state

Recall: Training Seq2Seq Model



- 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

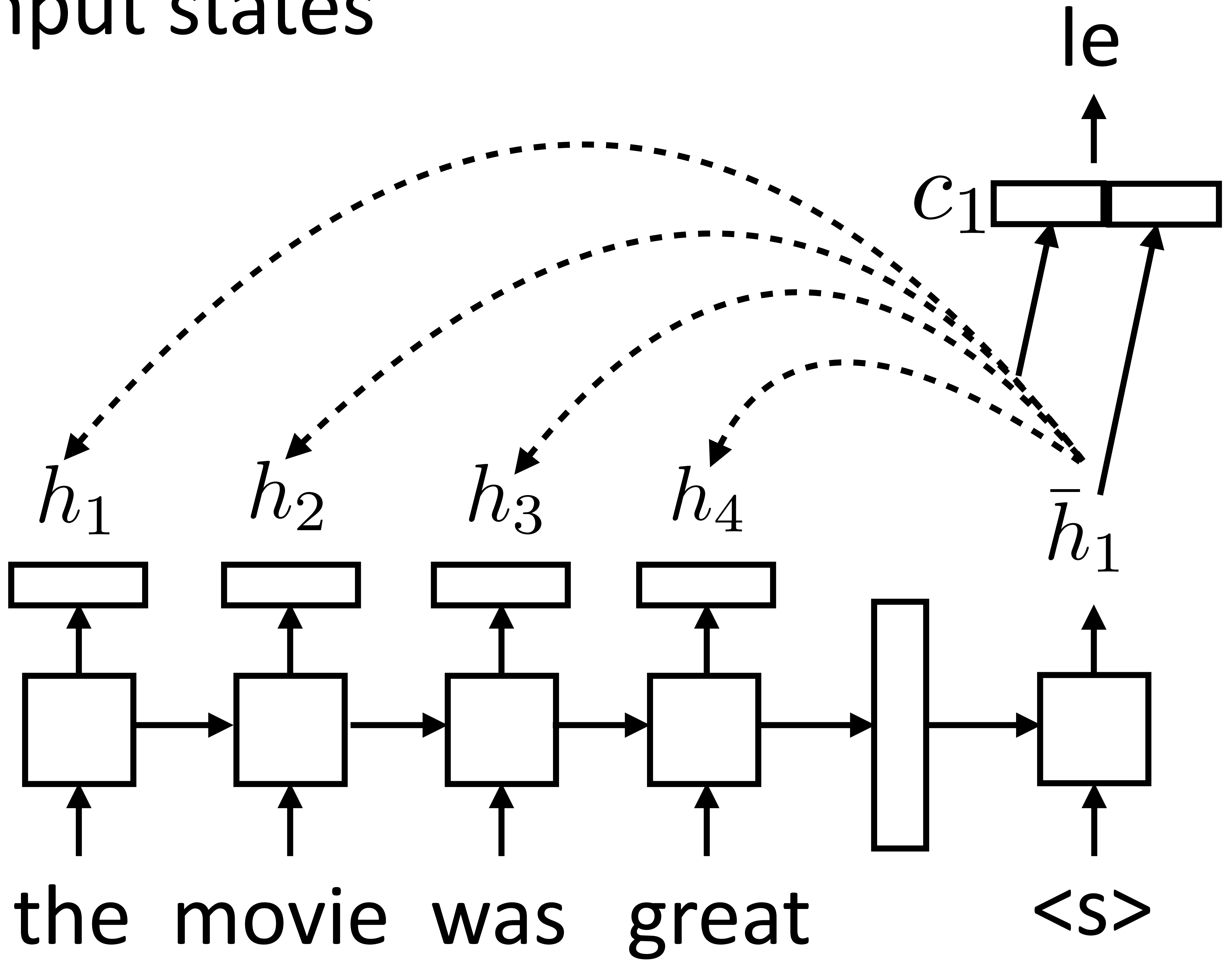
Recall: Attention



- 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



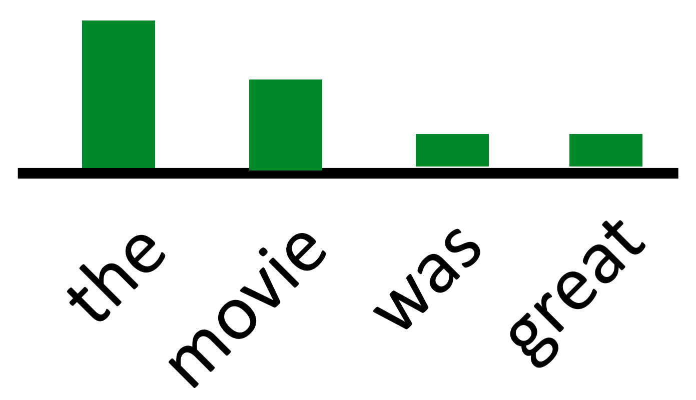
- No attn: $P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W\bar{h}_i)$

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

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

- Weighted sum of input hidden states (vector)

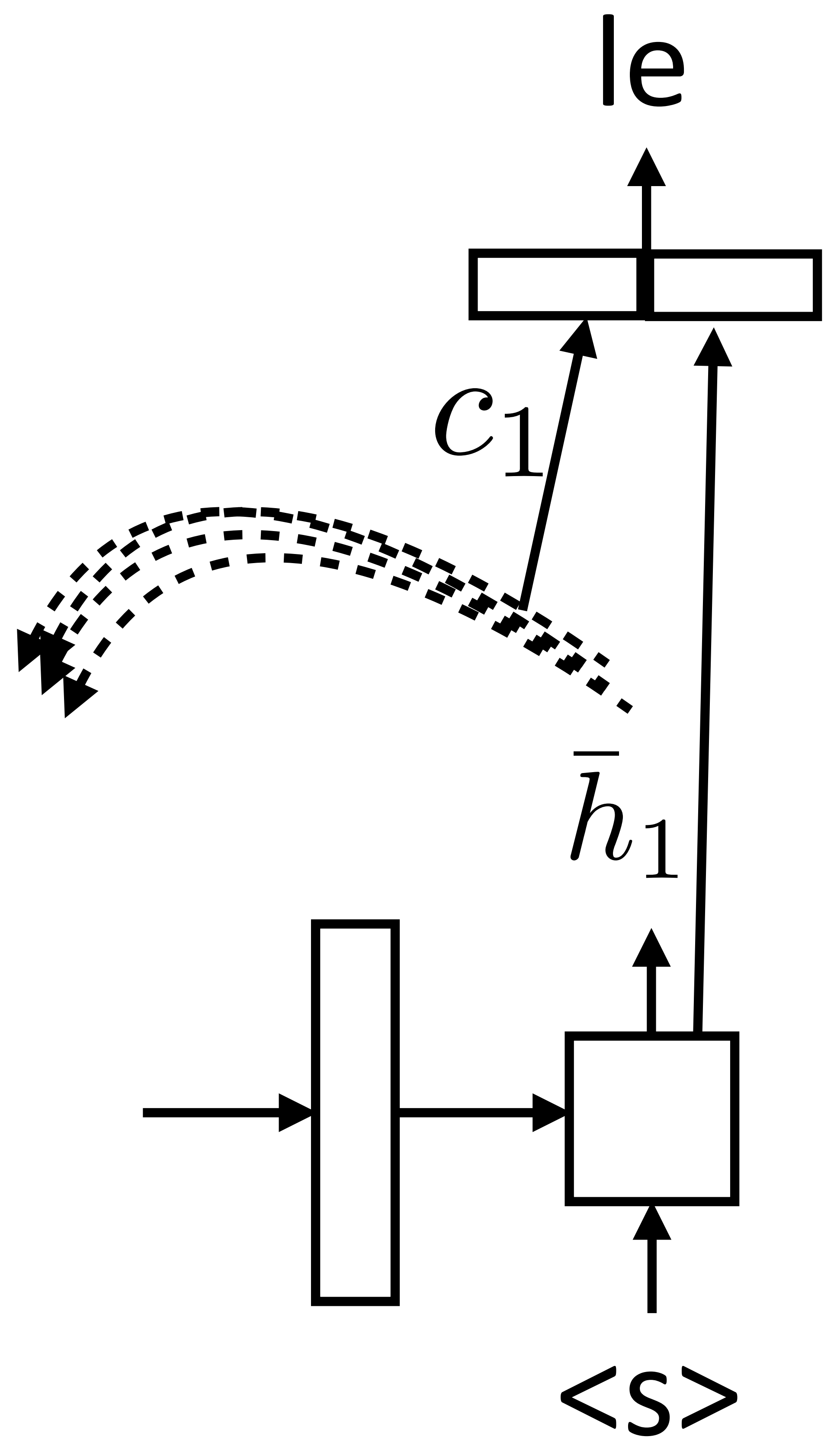
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$



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

- 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

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

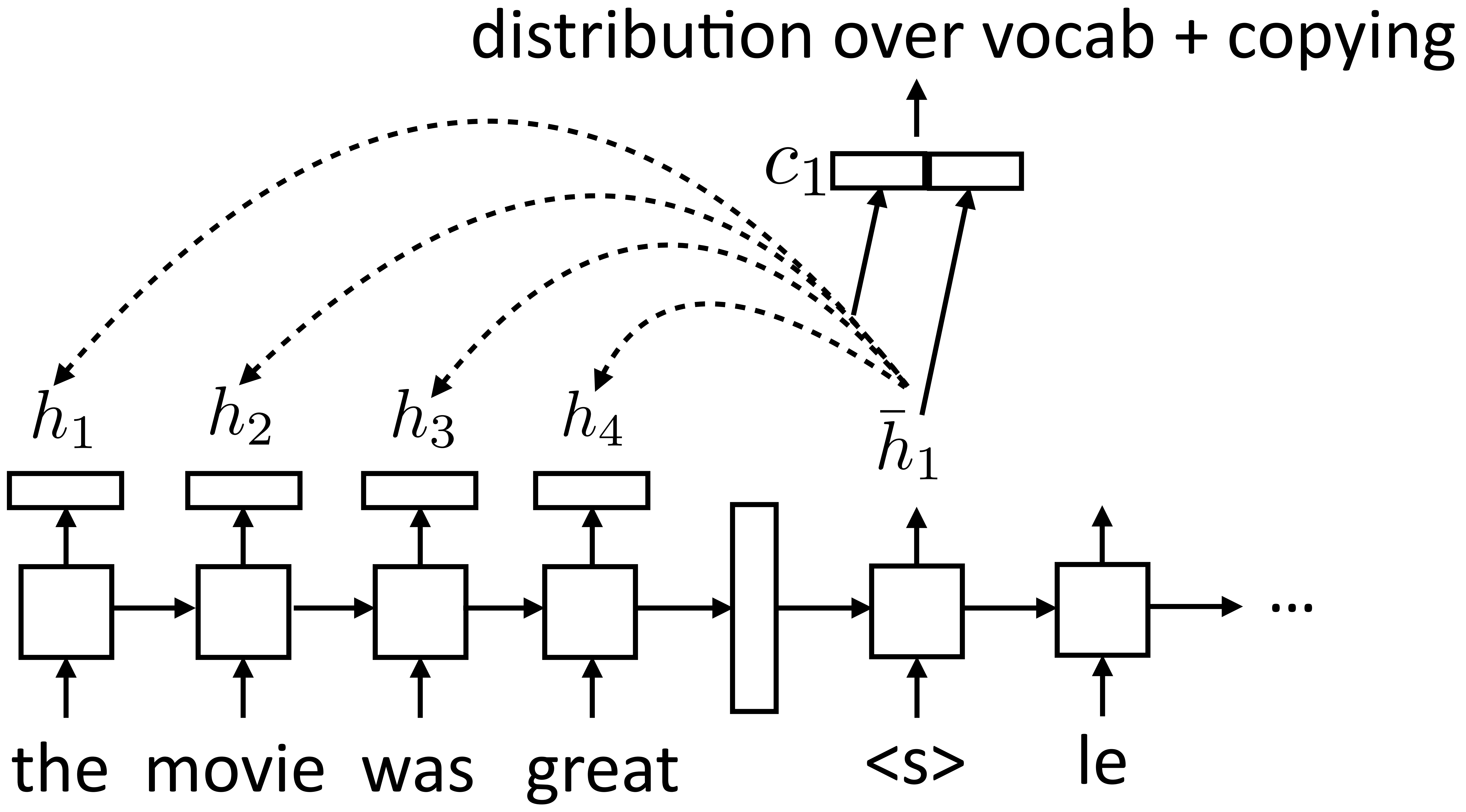
- Luong+ (2015): bilinear

- Note that this all uses outputs of hidden layers

Neural MT

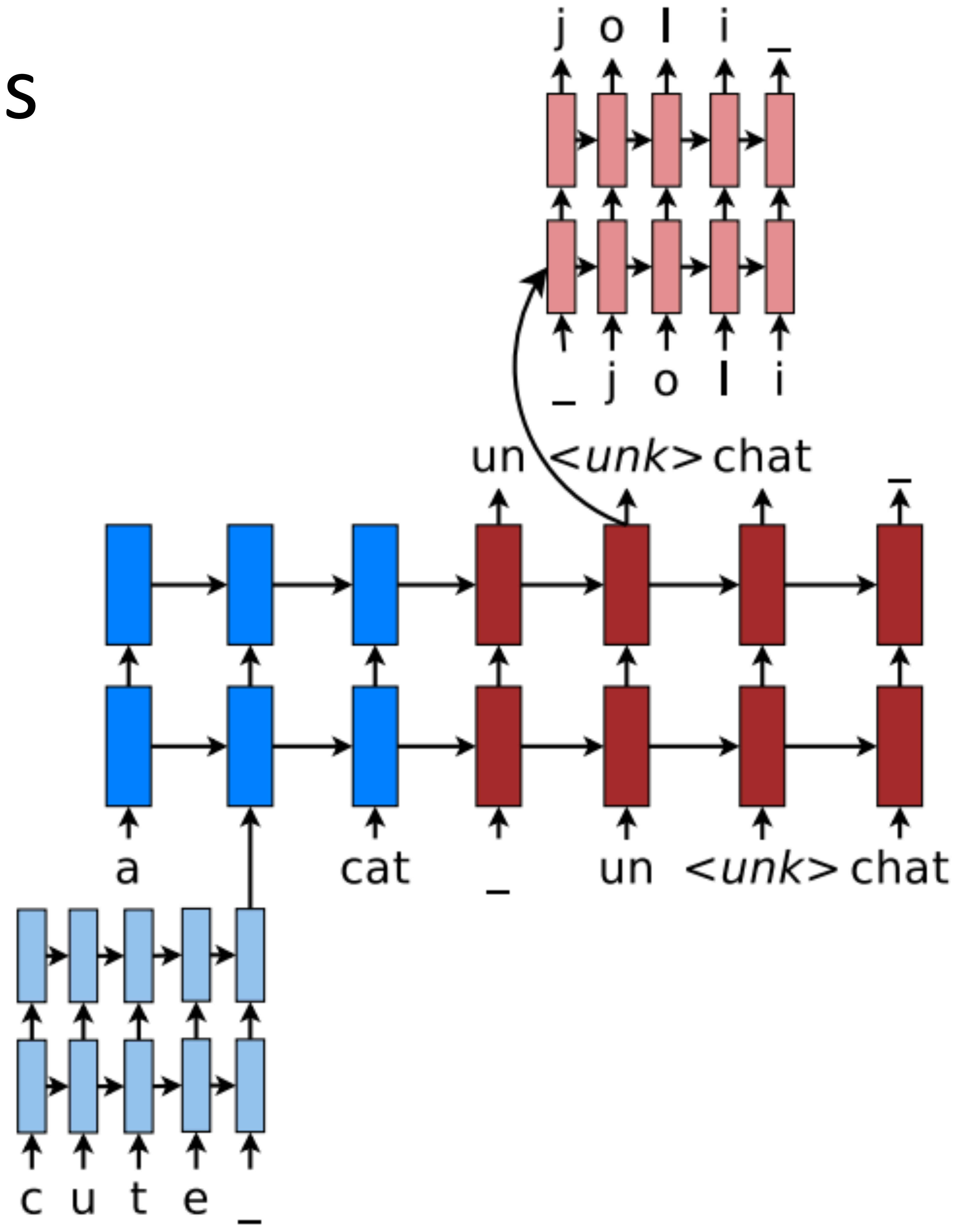
Encoder-Decoder MT

- encoder-decoder with attention and copying for rare words



Rare Words: Character Models

- If we predict an unk token, generate the results from a character LSTM
- Can potentially transliterate new concepts, but architecture is more complicated and slower to train
- Models like this in part contributed to dynamic computation graph frameworks becoming popular



Luong et al. (2016)

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

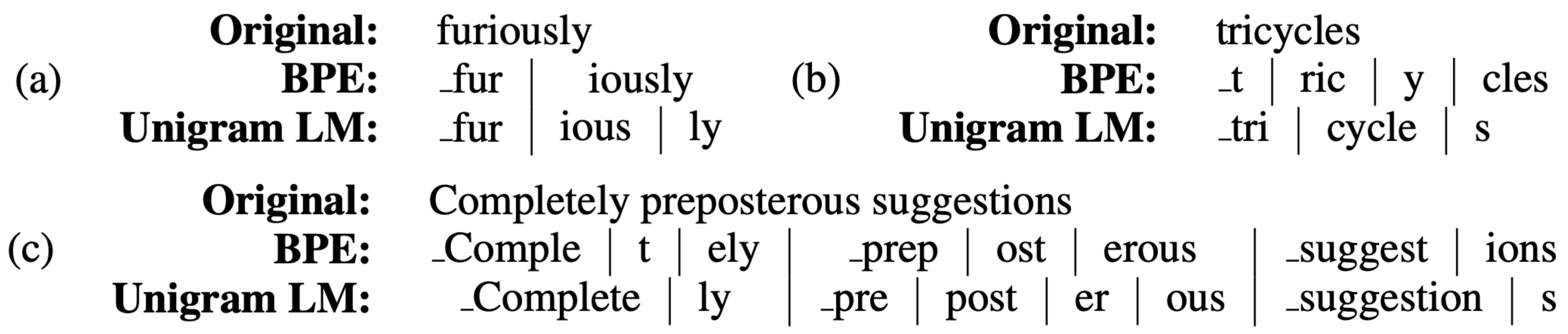
Sennrich et al. (2016)

Word Pieces

```
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

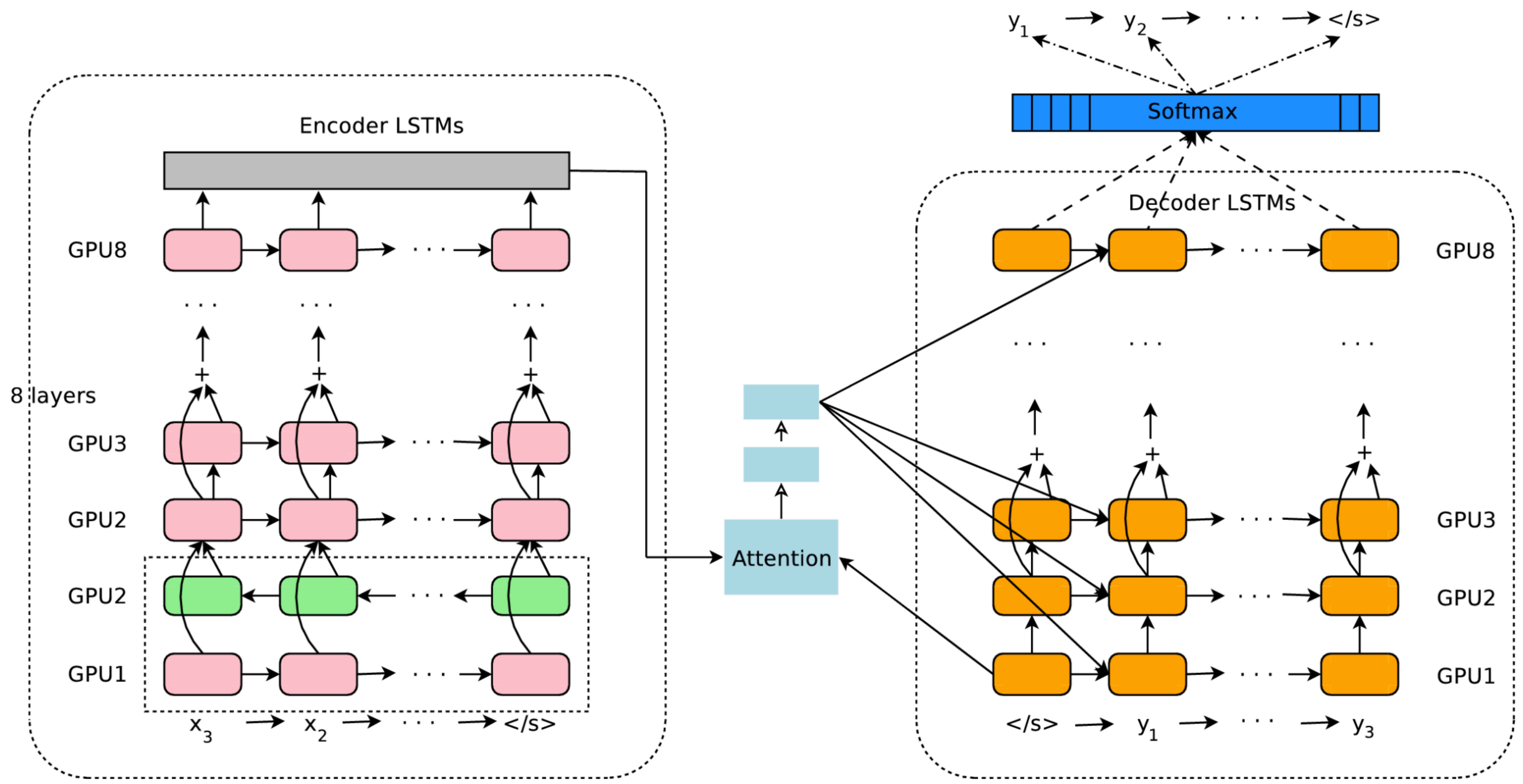


(a)	**Original:**	furiously
	BPE:	_fur | iously
	Unigram LM:	_fur | ious | ly
(b)	**Original:**	tricycles
	BPE:	_t | ric | y | cles
	Unigram LM:	_tri | cycle | s
(c)	**Original:**	Completely preposterous suggestions
	BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
	Unigram LM:	_Complete | 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's NMT System



y_1 → y_2 → · · · → </s>
Softmax
Encoder LSTMs
Decoder LSTMs
8 layers
GPU8
GPU3
GPU2
GPU2
GPU1
Attention
x_3 → x_2 → · · · → </s>
</s> → y_1 → · · · → y_3

- 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)

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)

Human Evaluation (En-Es)

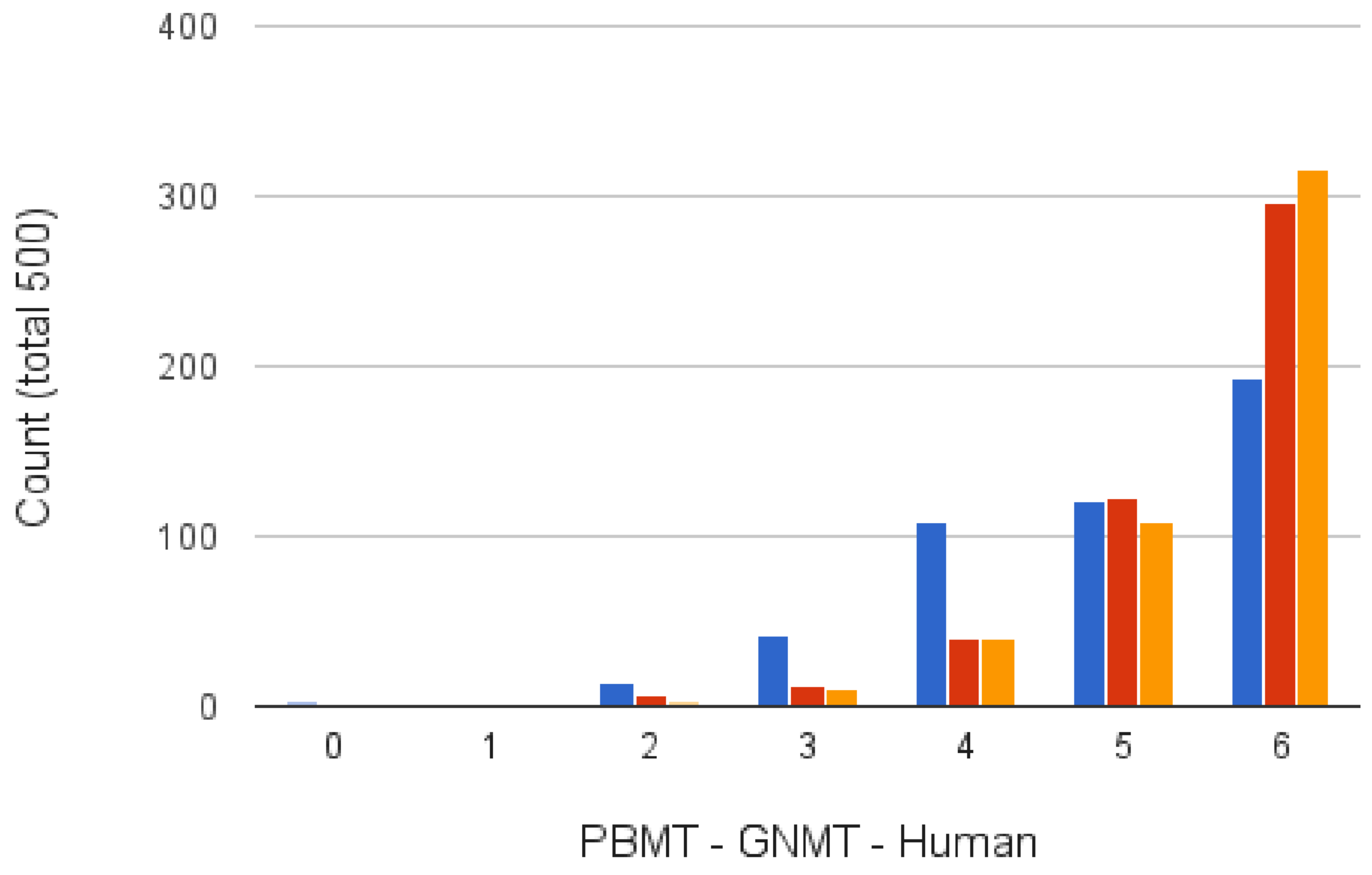


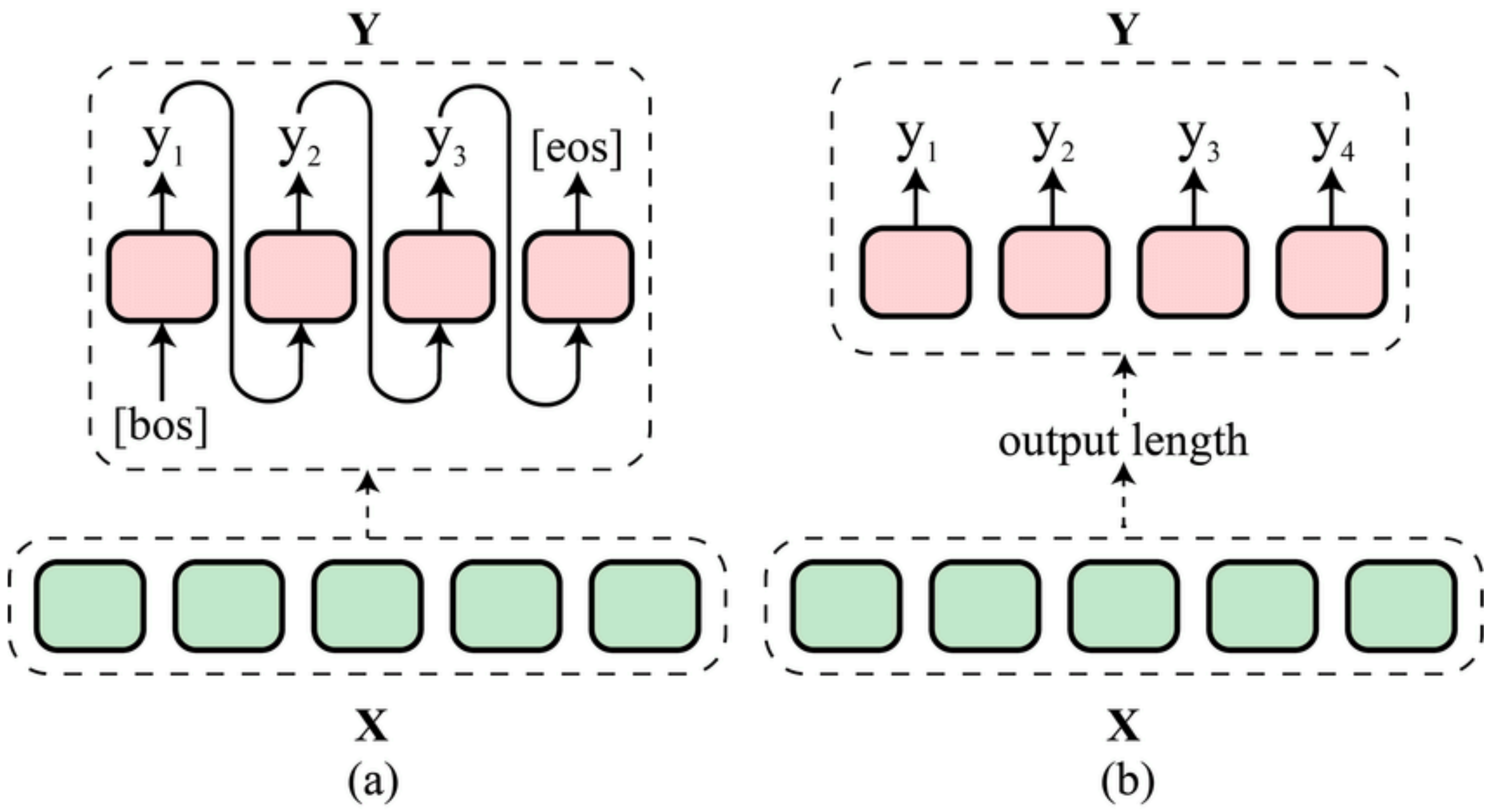
Figure 6: Histogram of side-by-side scores on 500 sampled sentences from Wikipedia and news websites for a typical language pair, here English → Spanish (PBMT blue, GNMT red, Human orange). It can be seen that there is a wide distribution in scores, even for the human translation when rated by other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

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

Wu et al. (2016)

Frontiers in MT

Non-Autoregressive NMT



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

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

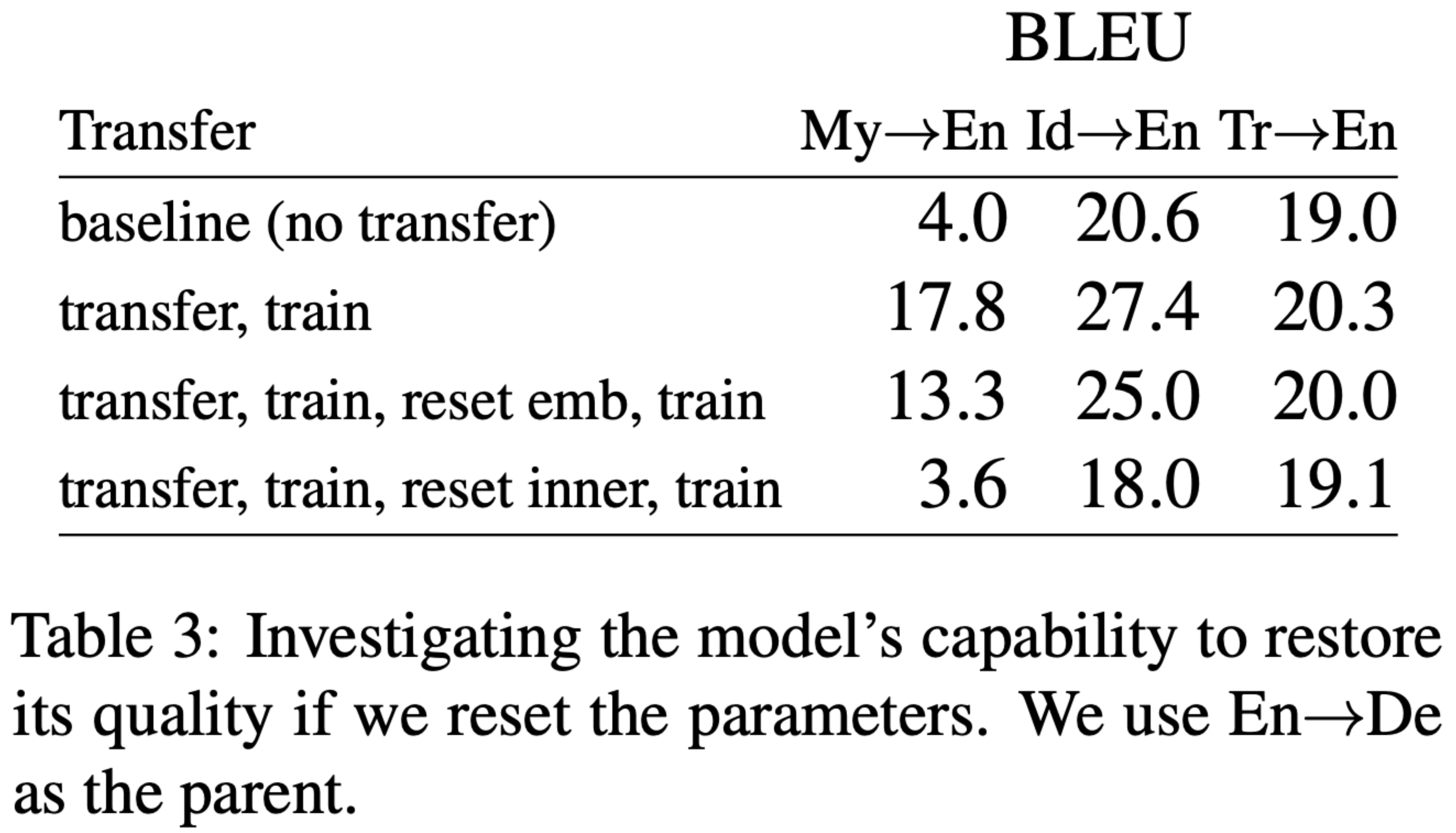
Low-Resource MT

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

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

- Pre-trained models can help further

Burmese, Indonesian, Turkish

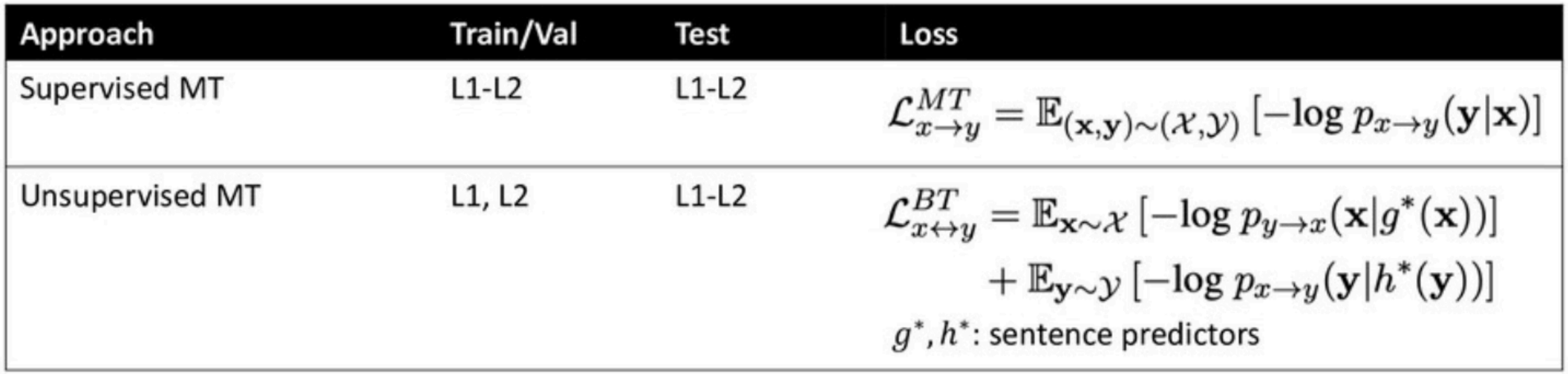


Transfer	BLEU My→En	Id→En	Tr→En
baseline (no transfer)	4.0	20.6	19.0
transfer, train	17.8	27.4	20.3
transfer, train, reset emb, train	13.3	25.0	20.0
transfer, train, reset inner, train	3.6	18.0	19.1

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)

Unsupervised MT



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