

Generative AI from First Principles

Lecture 3: Sequence Modeling

Translation and Attention

Lecture 3: Sequence Modeling, Translation, and Attention

Previously: language modeling, before attention. **Today:** attention, through machine translation

1. Sequence tasks and translation

sequence modeling; machine translation, how it is learned, decoded, and scored

2. Recurrent encoder-decoders

the encoder-decoder framework; models improved one after another; their drawbacks

3. Attention

its inception and its variants

4. Beyond recurrence

convolutional sequence models, translation from attention alone, self-attention and the Transformer

From Language Modeling to Sequence Modeling

Sequence tasks

sequences: text, speech, sensor readings; the length varies, and order matters: *dog bites man* $\neq$ *man bites dog*

language modeling: a distribution over sequences, one next token at a time

transduction: a sequence in, another out, of a different length and order

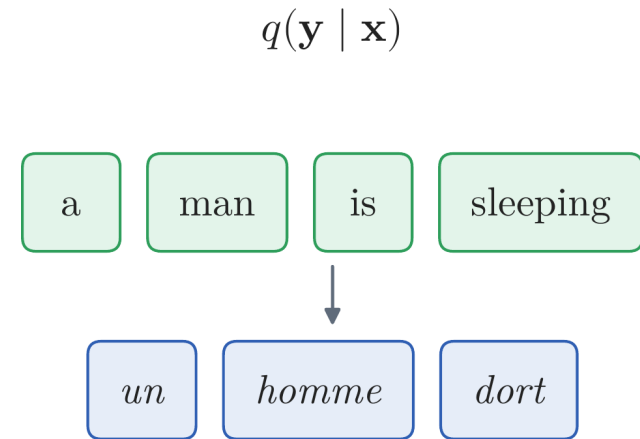
classification: one label for the whole sequence

labeling: one label for every position

forecasting: the future of a series, from its past

anomaly detection: how unusual each part is

the categories overlap: labeling and forecasting are transductions too; a forecaster flags anomalies



the output may differ in length and order

This lecture: translation, a transduction

From Language Modeling to Sequence Modeling

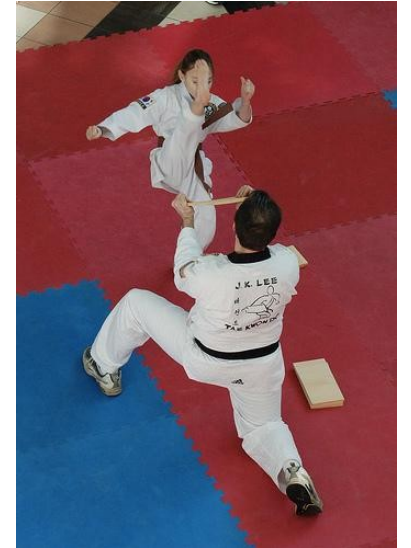
Translation

English *a girl in karate uniform breaking a stick with a front kick .*

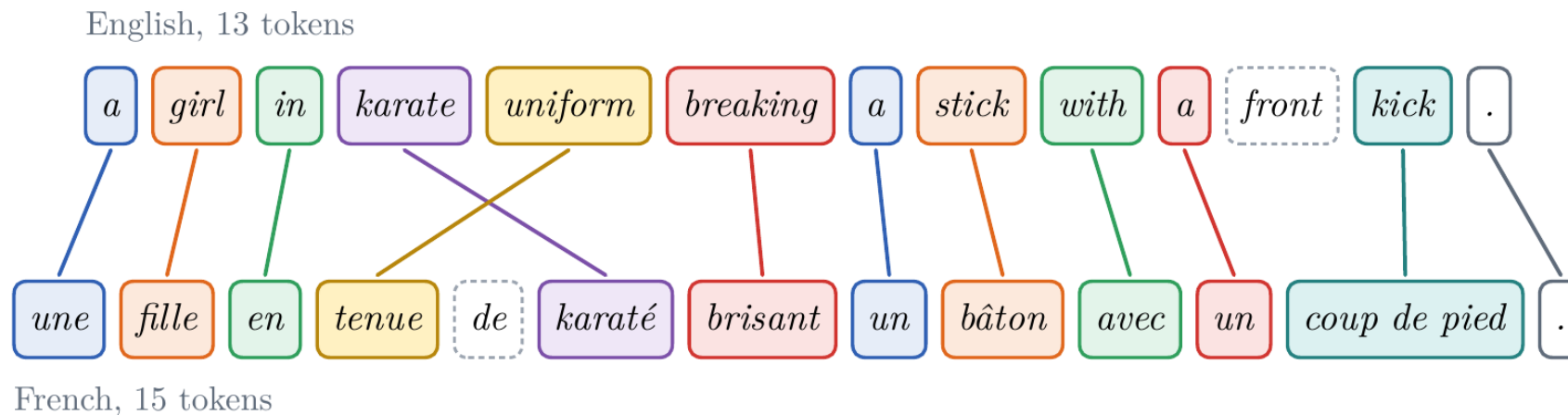
French *une fille en tenue de karaté brisant un bâton avec un coup de pied .*

translating one sequence into another is hard

- the lengths differ: 13 tokens become 15
- the alignment is unknown: nothing says which word comes from which
- the order changes: *karate uniform* becomes *tenue de karaté*



Flickr30k image 101362133



the alignment, drawn by hand: nothing in the data gives it

From Language Modeling to Sequence Modeling

Learning from a parallel corpus

a parallel corpus: sentences paired with their translations, made by people

only the pairs: never which word comes from which

Multi30k: English captions of Flickr photographs, translated into French

29,000 pairs to train, 1,014 to validate, 1,000 to test

our models read the text only; the photographs show what a caption is



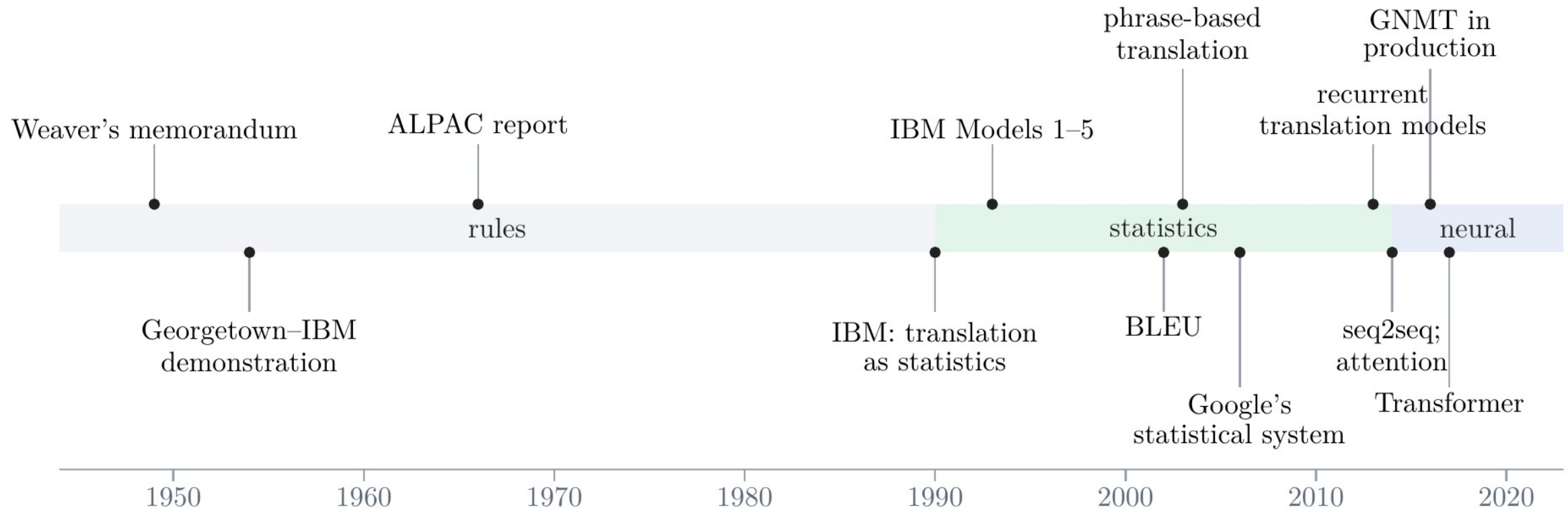
A man in an orange hat starring at something.

Un homme avec un chapeau orange regardant quelque chose.

Flickr30k image 1007129816, with its Multi30k caption

From Language Modeling to Sequence Modeling

Machine translation: rules, statistics, neural networks



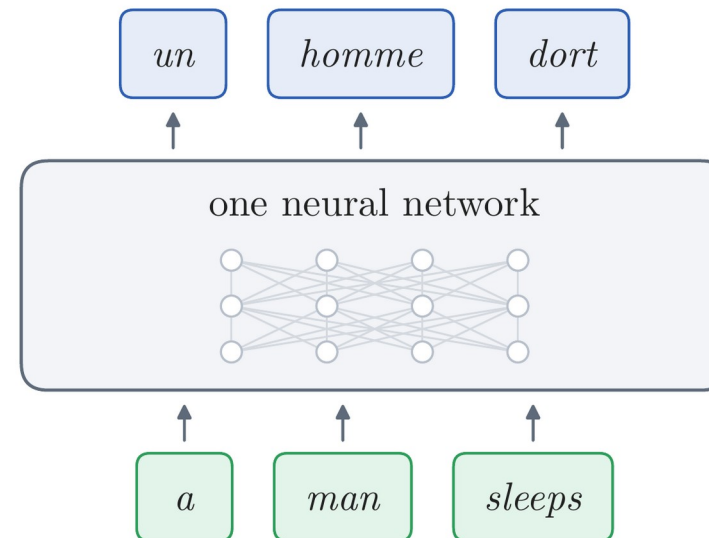
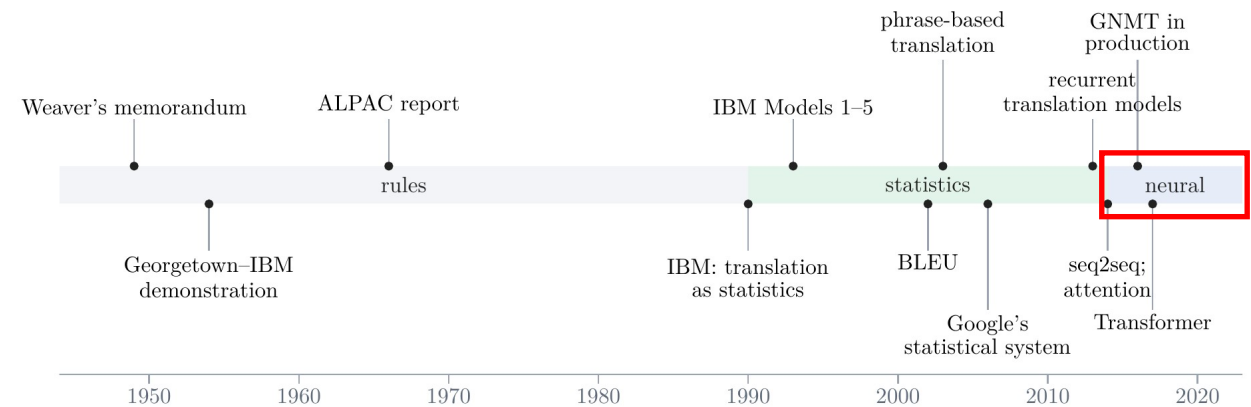
From Language Modeling to Sequence Modeling

Machine translation: rules, statistics, neural networks

rules (1949 on): dictionaries and grammar rules, written by hand; kept for narrow uses, such as weather bulletins

statistics (1990 on): learned from parallel corpora: which words translate which (the alignment, a hidden variable) and a language model of the target language; from 2003, phrase by phrase

neural networks (2014 on): one network for $q(y | x)$, trained end to end on the pairs; in production from 2016



$q_{\theta}(y | x)$,
trained end to end
on the pairs

Translation as Conditional Sequence Prediction

The model

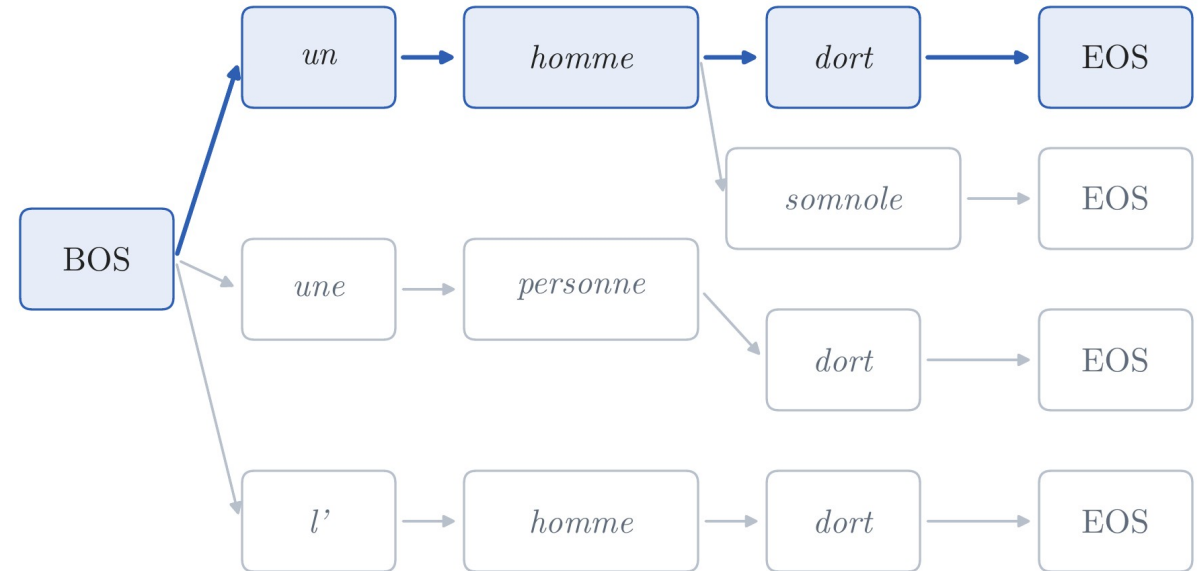
translation as a conditional language model: the chain rule of Lecture 2, with the source as extra context

$$q_{\theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{m+1} q_{\theta}(y_t \mid y_{<t}, \mathbf{x}), \quad y_{m+1} = \text{EOS}$$

EOS: the model decides where the translation stops

a tree of translations: every path from BOS to EOS is one; its probability is the product along the path

source: *a man sleeps*



Translation as Conditional Sequence Prediction

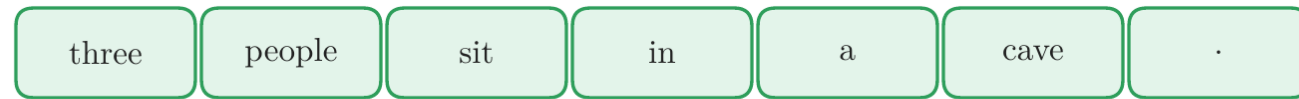
Training

negative log-likelihood: the log loss of Lecture 2, over the target tokens

$$\mathcal{L}(\theta) = -\frac{1}{P} \sum_{i=1}^P \sum_{t=1}^{m_i+1} \log q_{\theta}(y_t^{(i)} \mid y_{<t}^{(i)}, \mathbf{x}^{(i)})$$

teacher forcing: in training the decoder reads BOS and the reference, shifted by one, and predicts the next token at every position

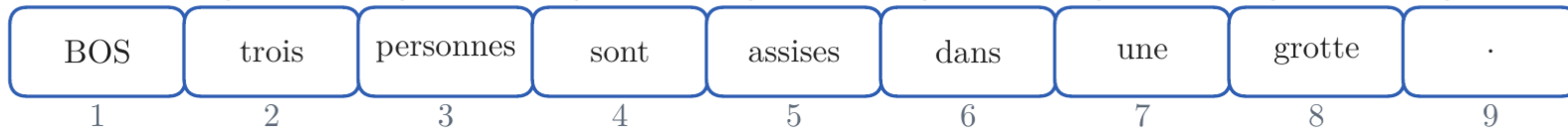
source
(read in full)



decoder
target



decoder
input
position



training: all inputs known • **generation**: each input is the word just chosen

Translation as Conditional Sequence Prediction

Decoding: greedy and beam search

the best translation: the most probable one

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} q_{\theta}(\mathbf{y} \mid \mathbf{x})$$

too many to try: $|V|$ choices of word at each of m steps,
 $|V|$ to the power m outputs

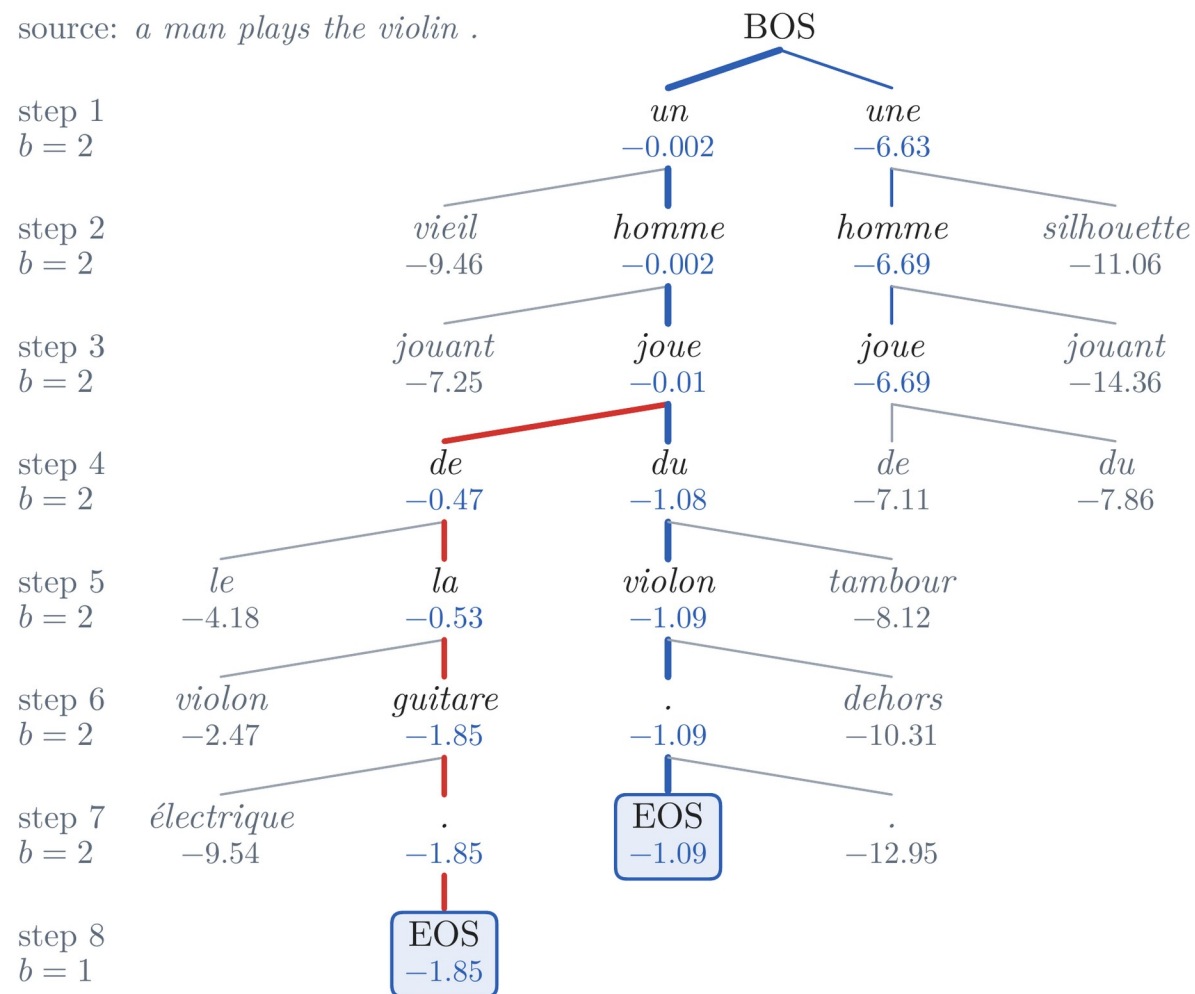
greedy: the most probable word at each step; a likely word now can lead to an unlikely sentence

step 4: *de* (-0.47) beats *du* (-1.08); greedy ends with *de la guitare*, -1.85

beam search: keep the B best partial translations, scored by total log-probability; greedy is $B = 1$

the beam keeps *du* and finishes *du violon*, -1.09: the right translation

source: *a man plays the violin .*



the attention model of Part 3 on a man plays the violin ., a beam of two

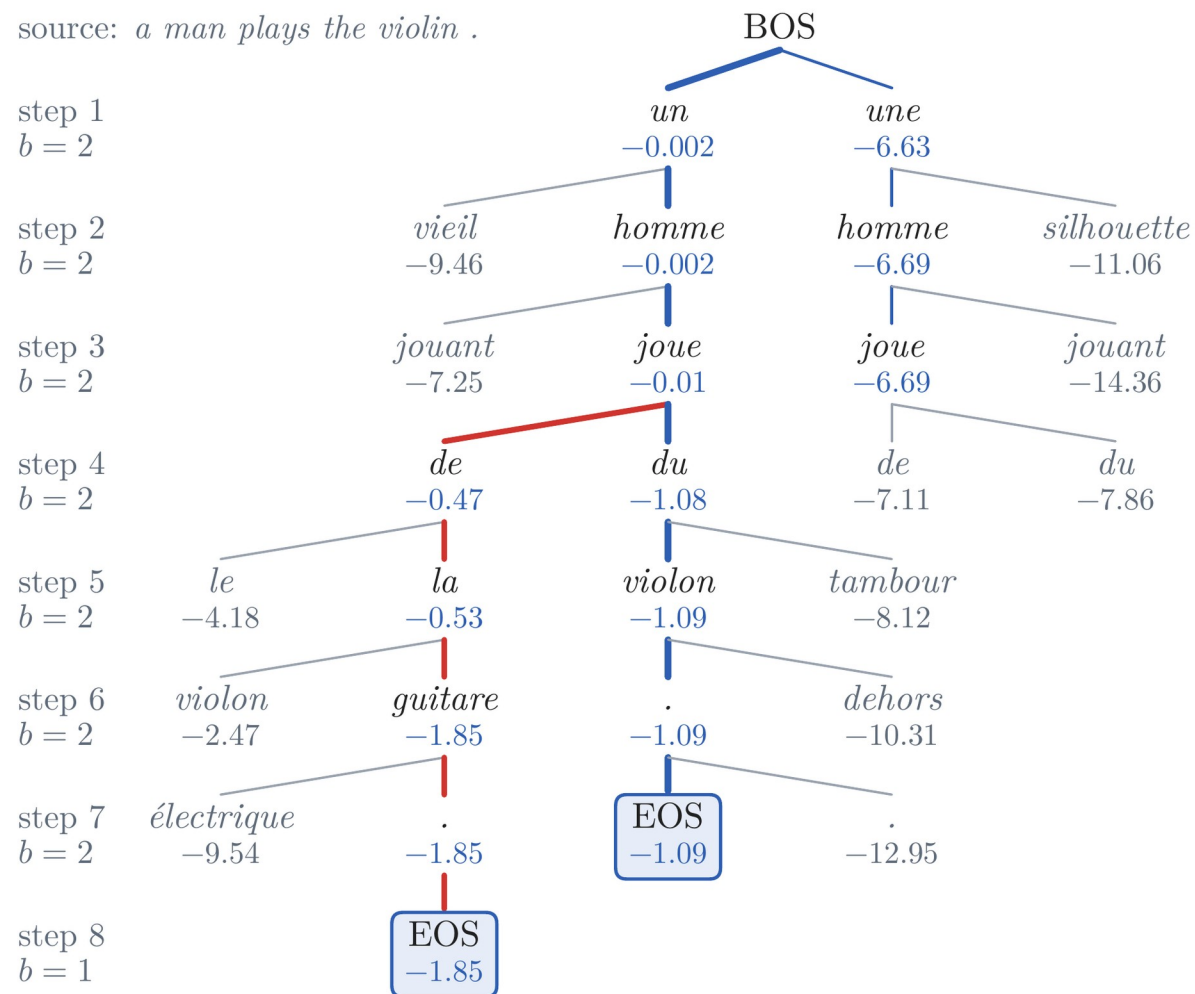
Translation as Conditional Sequence Prediction

Beam search: the algorithm

Beam search with a beam of size B

```
1 live ← {BOS}, score 0; finished ← {}; b ← B
2 while b > 0 do
3   extend each live hypothesis by its b likeliest tokens
4   keep the b best extensions: score = total log q
5   for each kept extension that ends with EOS do
6     move it to finished; b ← b - 1
7   end for
8   live ← the other kept extensions
9 end while
10 return the best of finished
```

source: *a man plays the violin .*



Translation as Conditional Sequence Prediction

Evaluation: BLEU

BLEU (Papineni et al., 2002) compares the output with the reference:

1. modified precision for $n = 1$ to 4: the share of the output's n -grams found in the reference, each counted at most as often as there

$$p_n = \frac{\sum_g \min(c_{\text{out}}(g), c_{\text{ref}}(g))}{\sum_g c_{\text{out}}(g)}$$

2. their geometric mean: each order weighs the same

$$\text{BLEU} = \text{BP} \cdot \exp\left(\frac{1}{4} \sum_{n=1}^4 \log p_n\right)$$

3. a brevity penalty: no reward for writing less

$$\text{BP} = \min(1, e^{1 - \ell_{\text{ref}} / \ell_{\text{out}}})$$

output	un	homme	joue	du	piano	
reference	un	homme	joue	du	violon	
$n = 1$	un	homme	joue	du	piano	found 4/5
$n = 2$	un homme	homme joue	joue du	du piano		3/4
$n = 3$	un homme joue	homme joue du	joue du piano			2/3
$n = 4$	un homme joue du	homme joue du piano				1/2

illustrative: a man plays the violin, with one wrong word

Translation as Conditional Sequence Prediction

Evaluation: the example worked

precisions: 4/5, 3/4, 2/3, 1/2; one wrong word spoils every n-gram that contains it

same length: BP = 1

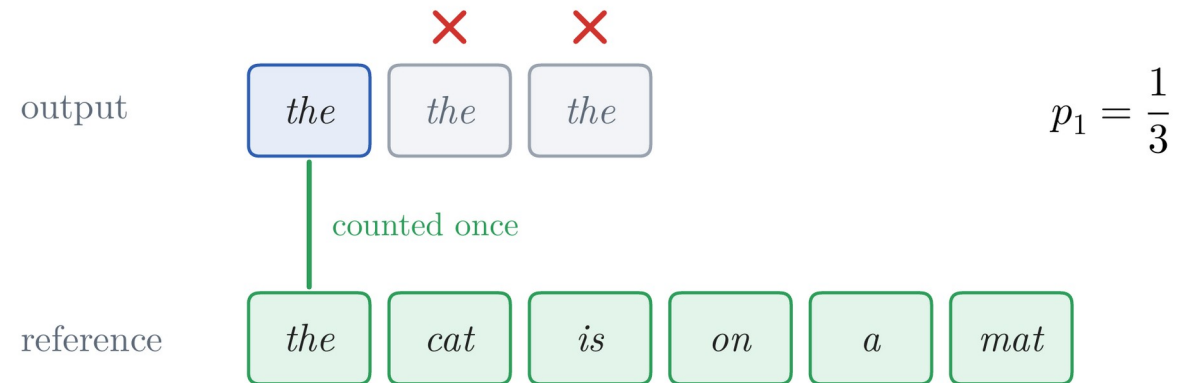
$$\text{BLEU} = 1 \cdot \left(\frac{4}{5} \cdot \frac{3}{4} \cdot \frac{2}{3} \cdot \frac{1}{2} \right)^{1/4} = 0.67$$

reported as **66.9**

clipping: each n-gram counted at most as often as in the reference: *the the the* earns one match

a real output: our attention model on the karate caption, *faisant* for *brisant*: 14/15, 12/14, 10/13, 8/12, BLEU 80.0

counts are summed over a whole test set before dividing



each *n*-gram counted at most as often as it occurs in the reference

Translation as Conditional Sequence Prediction

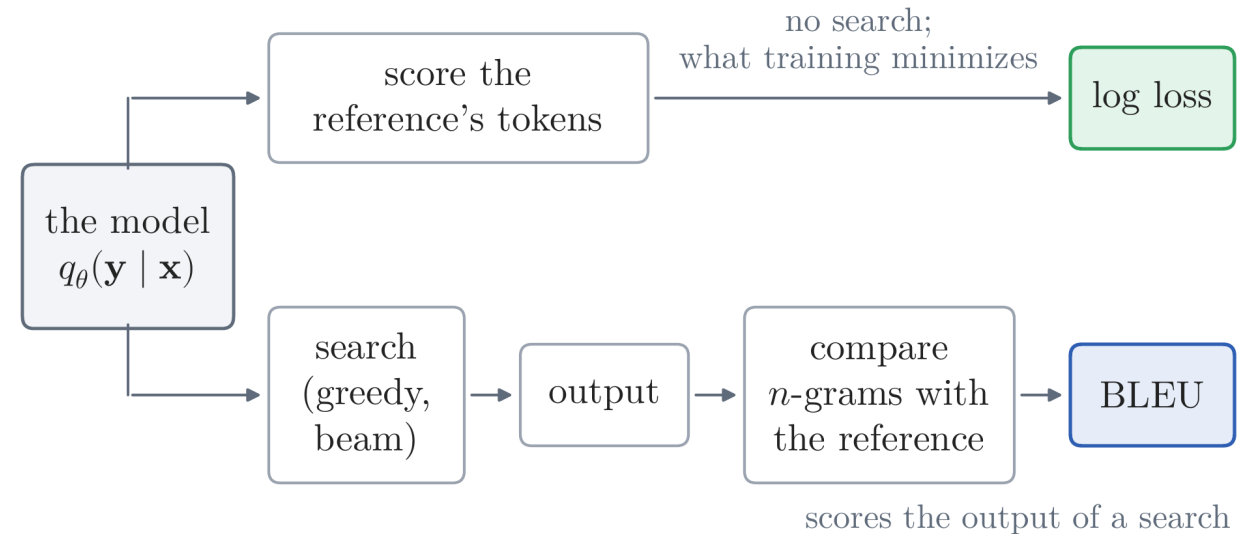
Evaluation: what BLEU measures

the log loss: the model's probabilities of the reference's tokens; no search

BLEU: the output of a search; a better search raises it, the model unchanged

it rewards matching words, not meaning: *une jeune fille* for *une fille* loses points

today: learned metrics such as COMET agree far better with people; BLEU stays here to compare the papers of 2014–2017



Lecture 3: Sequence Modeling, Translation, and Attention

1. Sequence tasks and translation

sequence modeling; machine translation, how it is learned, decoded, and scored

2. Recurrent encoder-decoders

the encoder-decoder framework; models improved one after another; their drawbacks

3. Attention

its inception and its variants

4. Beyond recurrence

convolutional sequence models, translation from attention alone, self-attention and the Transformer

Encoder-Decoders

Two roles

every translator of this lecture splits the work between two networks

encoder: reads the source once and returns its states Z

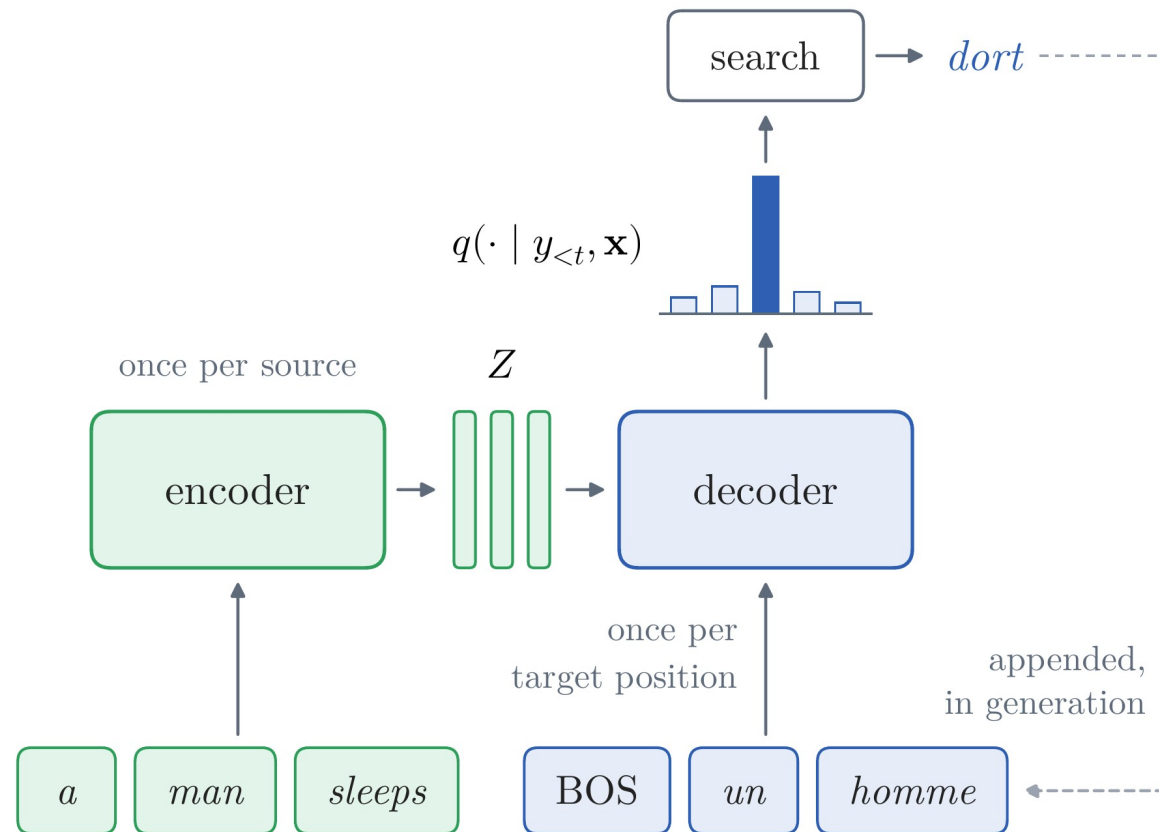
$$Z = \text{enc}(x_1, \dots, x_n)$$

decoder: from Z and the target prefix, the distribution of the next word; a search picks the word

$$q(\cdot \mid y_{<t}, \mathbf{x}) = \text{dec}(Z, y_0, \dots, y_{t-1})$$

roles, not architectures: each can be recurrent, convolutional, or attention

every landmark translator of 2013–2017 has these two roles, from Cho et al. (2014a) to the Transformer (2017); we meet each where its model comes in



Encoder-Decoders

The recurrent encoder-decoder

1. the **encoder** reads the source one token at a time; its last state is the context c

$$\mathbf{h}_j = f_{\text{enc}}(\mathbf{h}_{j-1}, E(x_j)), \quad \mathbf{c} = \mathbf{h}_n$$

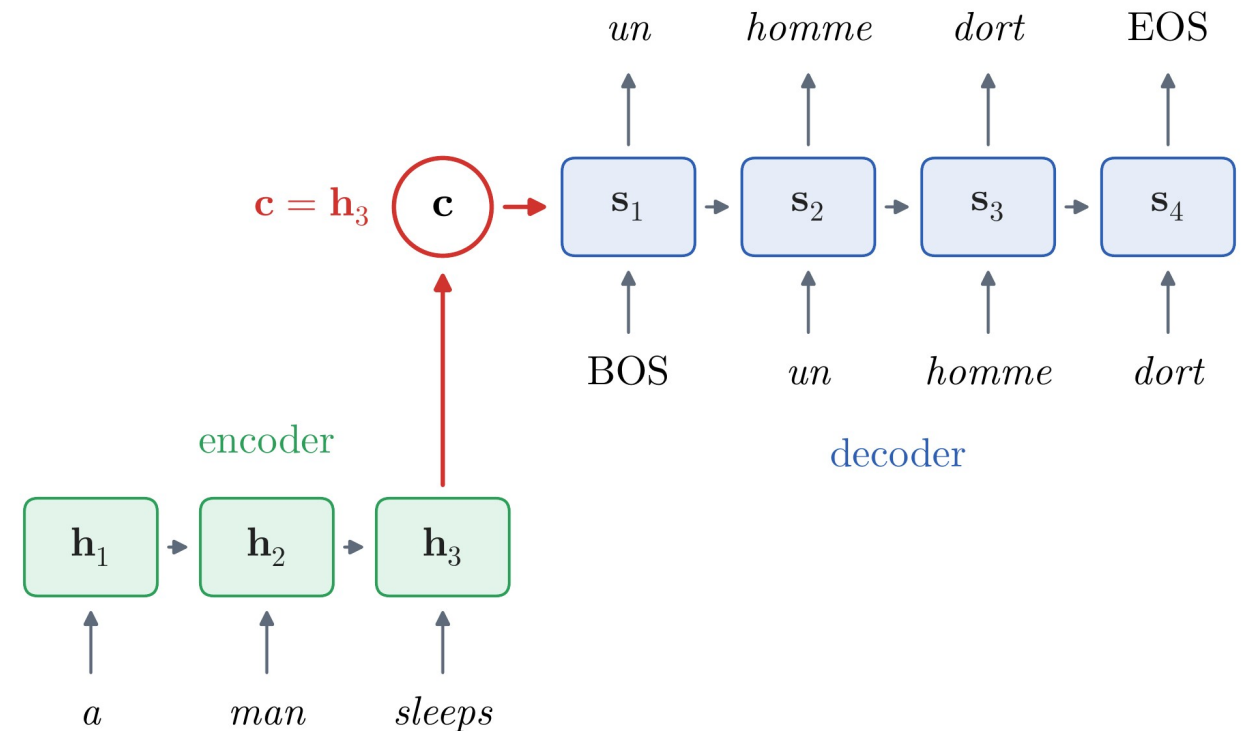
2. the **decoder** starts from c and reads the target prefix

$$\mathbf{s}_t = f_{\text{dec}}(\mathbf{s}_{t-1}, E(y_{t-1})), \quad \mathbf{s}_0 = \mathbf{c}$$

3. each **decoder state** gives the next word's distribution

$$q(y_t \mid y_{<t}, \mathbf{x}) = \text{softmax}(\mathbf{s}_t U)$$

trained end to end: the encoder learns to put into c what the decoder needs



Encoder-Decoders

The context at every step

the problem: the decoder sees the source once, at the start → **the fix here:** give c to every step

Cho et al. (2014a), the “RNN Encoder-Decoder”: c enters every decoder update and the output

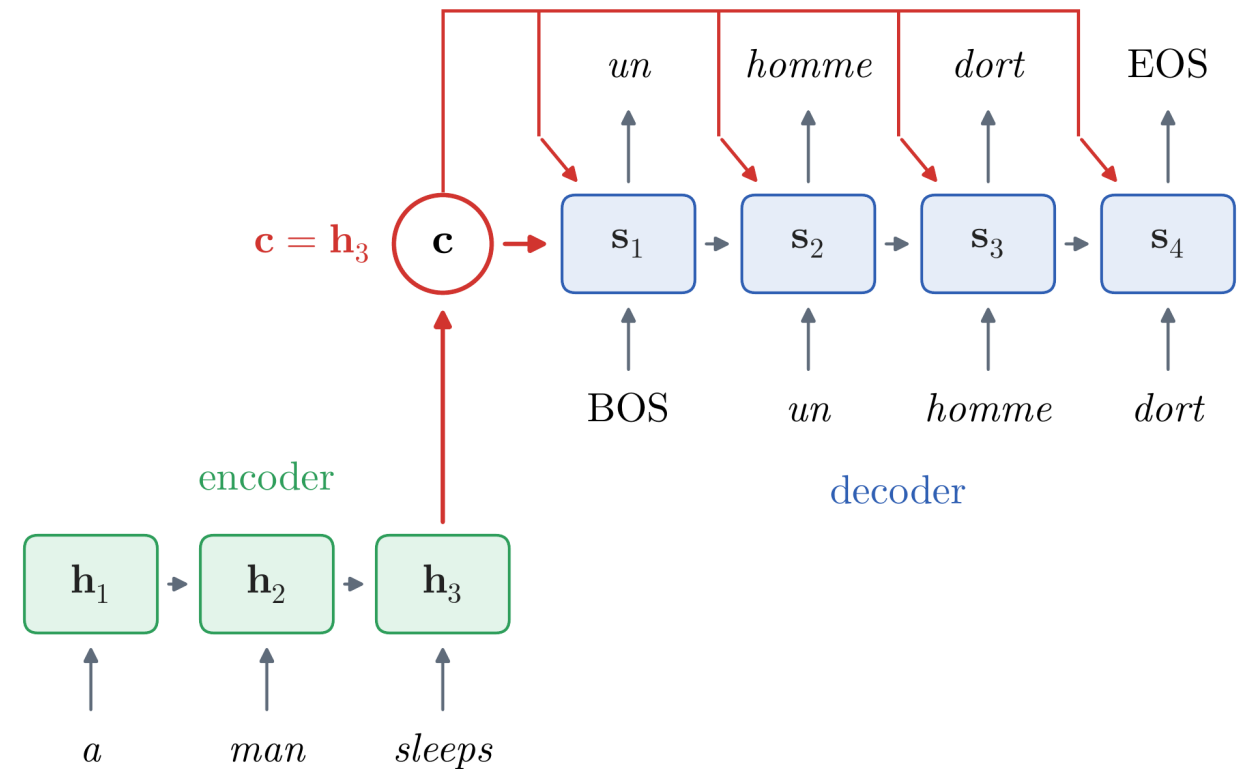
$$\mathbf{s}_t = f_{\text{dec}}(\mathbf{s}_{t-1}, E(y_{t-1}), \mathbf{c})$$

$$q(y_t | y_{<t}, \mathbf{x}) = \text{softmax}([\mathbf{s}_t; \mathbf{c}] U)$$

every prediction is one step from c , however long the prefix

used to score phrase pairs in a phrase-based system:
BLEU 33.30 → 33.87

still one summary for the whole source: 5 words or 50,
the same number of numbers (Part 3 removes it)



Encoder-Decoders

Recurrent cells

the problem: a plain RNN forgets along a long sentence → **the fix here:** gated cells, LSTM and GRU

a plain RNN: rewrites its state through tanh at every step; information and gradients fade over a long sentence

$$\mathbf{h}_t = \tanh(\mathbf{h}_{t-1}W + \mathbf{e}_tV + \mathbf{b})$$

LSTM: a memory cell only scaled by a gate and added to

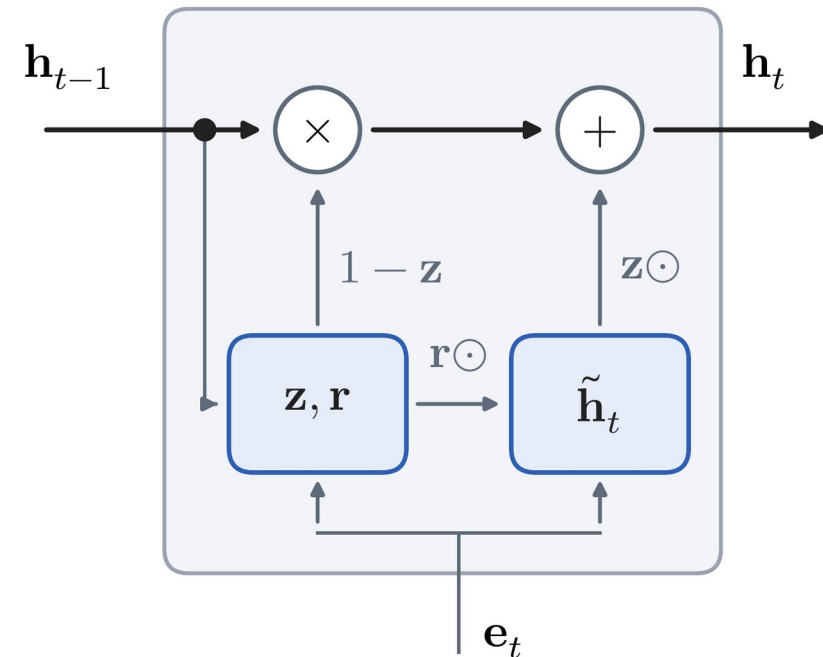
$$\mathbf{m}_t = \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

GRU (Cho et al., 2014a): two gates; the state is the memory

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

the roles, the training and the context stay the same

(c) GRU



hands over $\mathbf{h}_n$

Encoder-Decoders

A bidirectional encoder, a causal decoder

the problem: a left-to-right state knows only the words before it → **the fix here:** read the source both ways

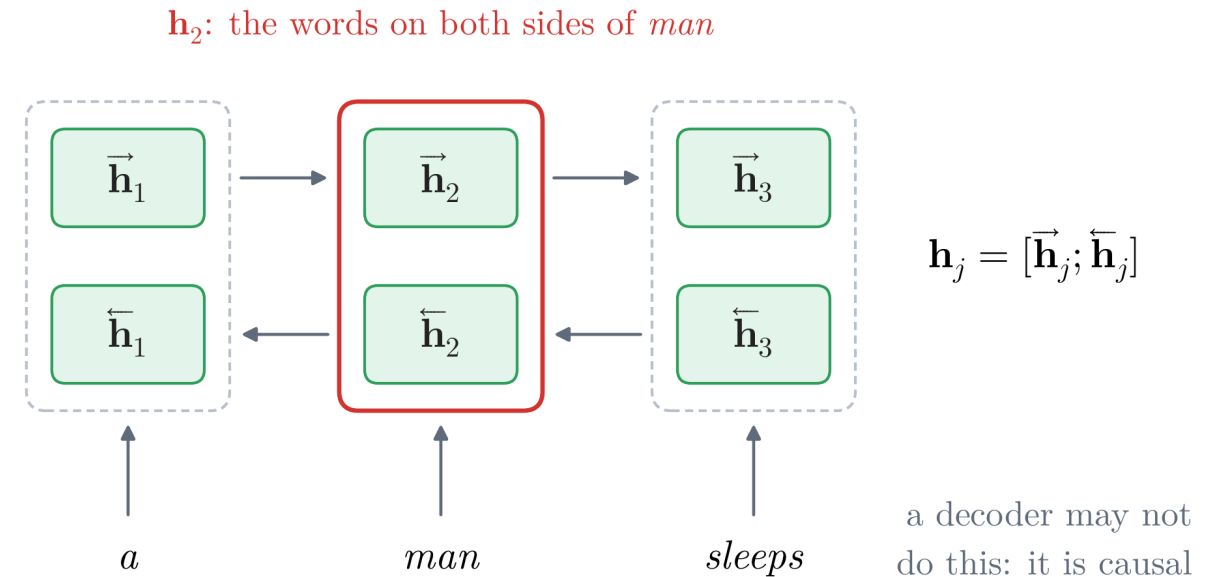
the source is given in full before translation begins

bidirectional encoder: one RNN left to right, one right to left; join their states

$$\mathbf{h}_j = [\vec{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$$

the state of each word summarizes **the whole sentence around it**

causal decoder: each prediction reads only the words before it; in training a decoder that read ahead would copy its answer



Sequence to Sequence: a Complete Neural Translator

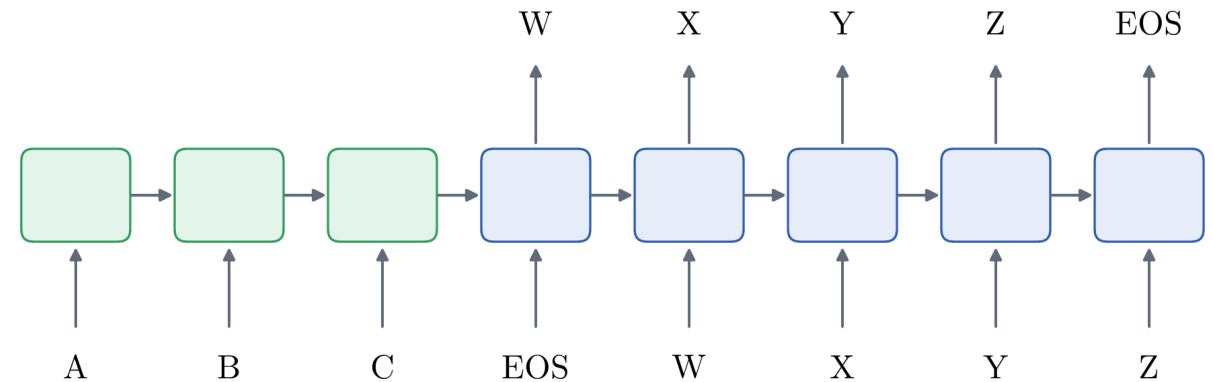
Sutskever et al. (2014): the whole translator

one network is the whole translator, trained on sentence pairs alone

encoder and decoder: deep LSTMs

the final states of the encoder start the decoder: the only route from the source

two choices made it work: depth, and reading the source backwards



encoder reads the source

decoder writes the target

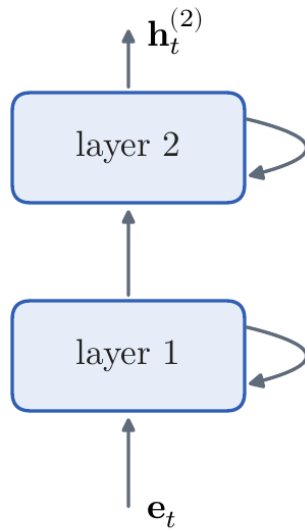
redrawn from Sutskever et al. (2014), Fig. 1: read ABC, then write WXYZ

Sequence to Sequence: a Complete Neural Translator

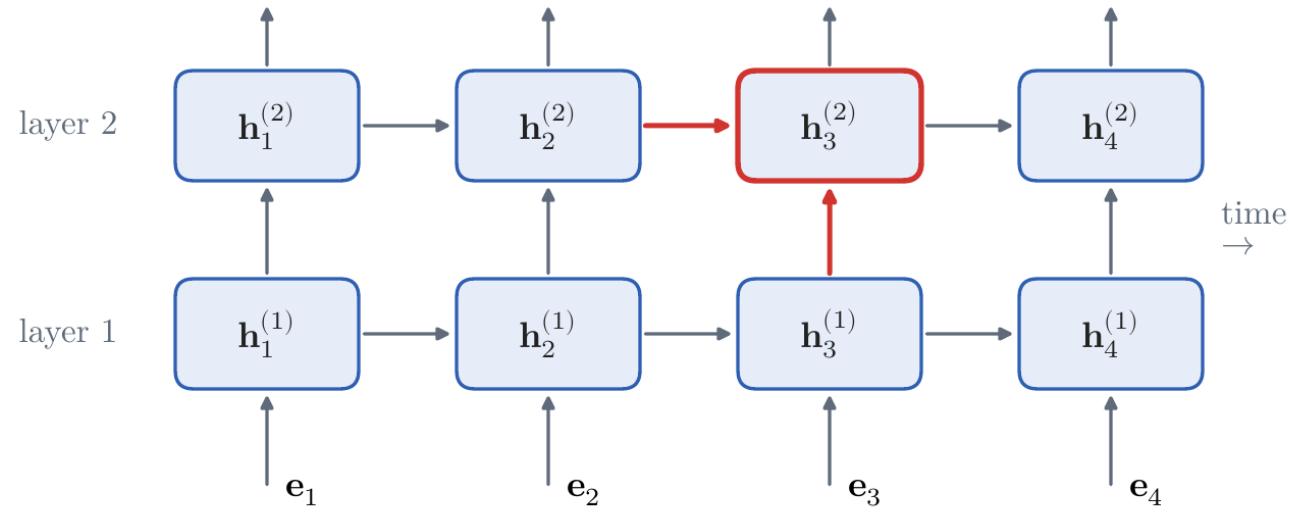
The model: deep recurrent networks

the problem: one layer is shallow → **the fix here:** stack layers: four, in Sutskever et al.

(a) rolled up



(b) unrolled over four steps



layer ℓ at step t reads the layer below at step t , and itself at step $t - 1$

Sutskever et al.: each further layer lowered the perplexity by nearly 10%

Sequence to Sequence: a Complete Neural Translator

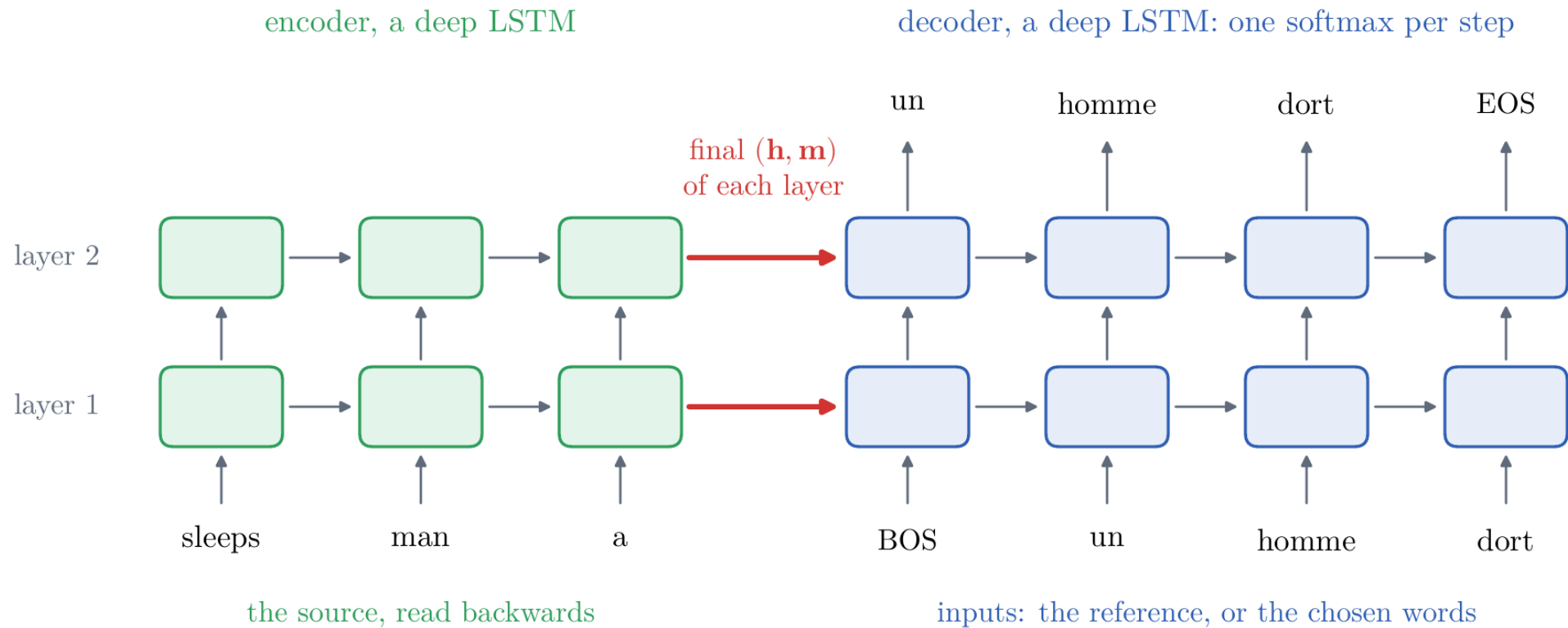
The model: two deep LSTMs

two deep LSTMs: four layers of 1,000 units each

vocabularies: 160,000 English and 80,000 French words

the final states of the encoder start the decoder

no attention: the source arrives only through them

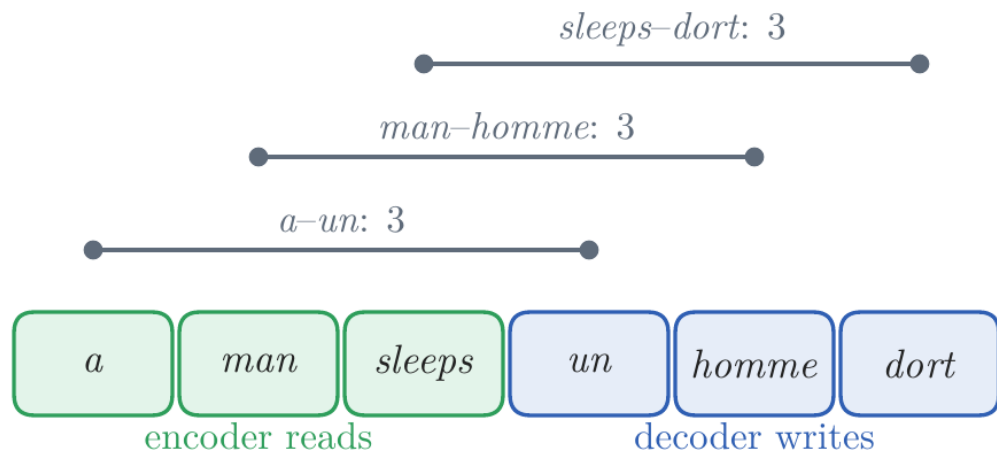


Sequence to Sequence: a Complete Neural Translator

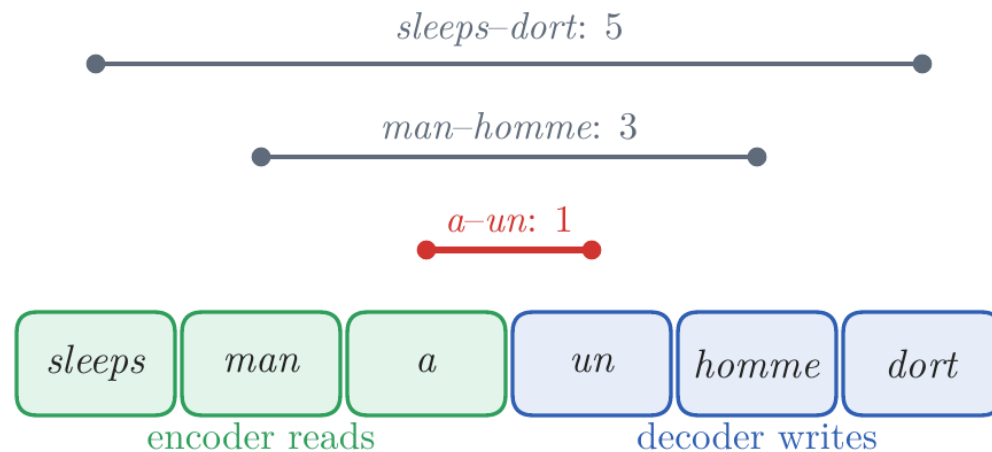
Reading the source backwards

the problem: the first words of the two sentences are far apart → **the fix here:** read the source backwards

(a) the source in its order



(b) the source read backwards



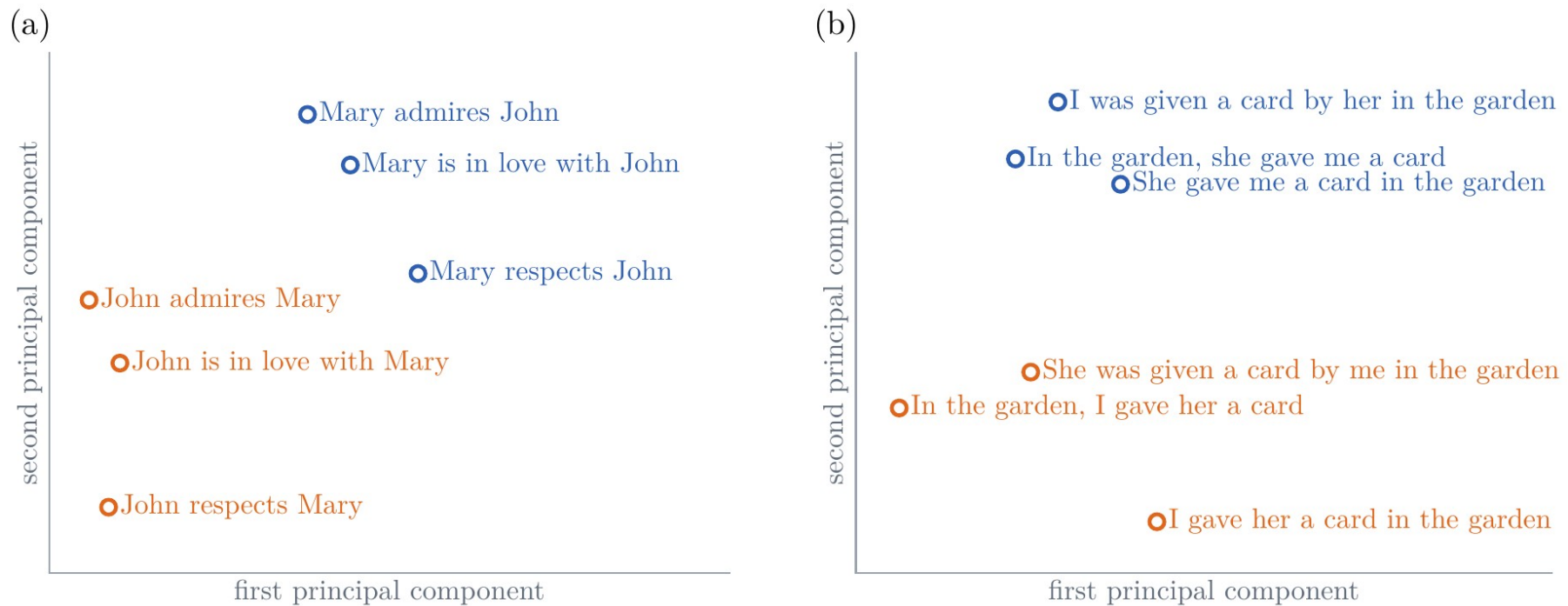
found by experiment: Sutskever et al.'s single model, source read forwards, then backwards: perplexity 5.8 → 4.7, BLEU 25.9 → 30.6

the same average distance between a word and its translation, but the first words are now adjacent with these short paths, backpropagation "has an easier time" (Sutskever et al., 2014)

Sequence to Sequence: a Complete Neural Translator

What the summary holds

the experiment: encode a few phrases; project the final states onto two dimensions (PCA)



redrawn from Sutskever et al. (2014), Fig. 2

word order matters: *John admires Mary* lands far from *Mary admires John*

voice matters less: *She gave me a card* lands near *I was given a card by her*

Sequence to Sequence: a Complete Neural Translator

Results

the task: WMT 2014 English–French, 12 million sentence pairs; BLEU on the test set

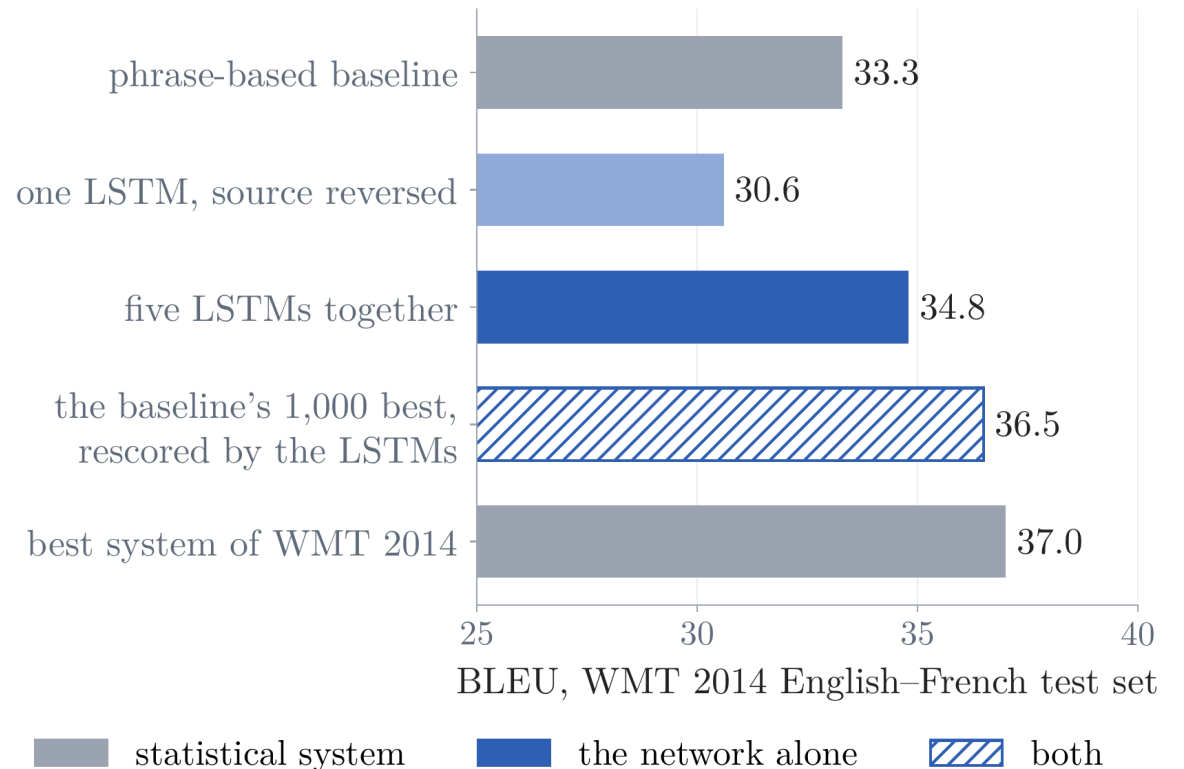
the model: 384 million parameters, about ten days on eight GPUs

the network alone: five LSTMs together, beam of 12: **34.8**, above their phrase-based baseline's 33.3

both: the baseline's 1,000 best translations, rescored by the LSTMs: 36.5; the best system of WMT 2014, a statistical one, listed in their paper: 37.0

“did not suffer on very long sentences”

from pairs alone, it beat a phrase-based system



every number from Sutskever et al. (2014)

Three Drawbacks of the Recurrent Encoder-Decoder

The fixed-size summary

our own experiment (not from the papers): write a string of 1 to 40 random letters backwards

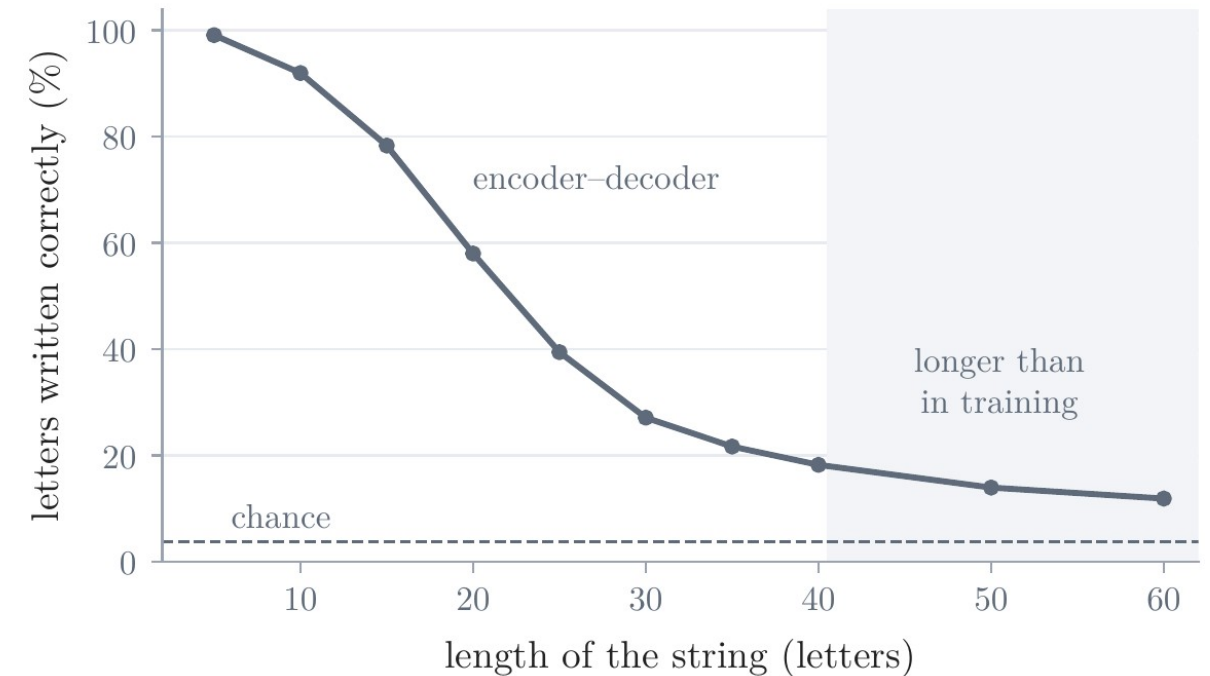
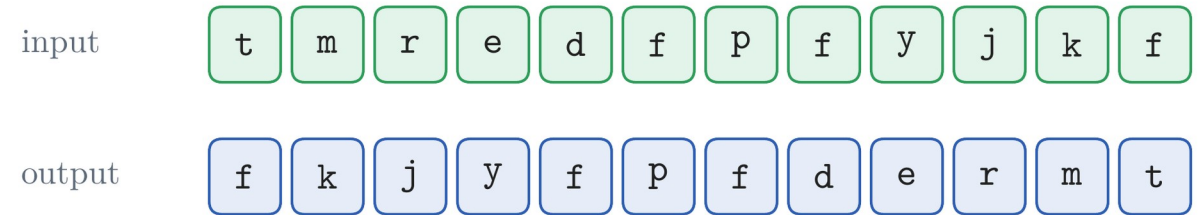
it needs memory and no knowledge of language

encoder-decoder: bidirectional LSTM encoder, LSTM decoder started from its final states; 4,000 steps of 64 random strings

measured: letters written in the right place

10 letters: 92%; 20: 58%; 40: 18%

access to the source: the longer the string, the less room per letter



ours: the encoder-decoder on the reversal task

Three Drawbacks of the Recurrent Encoder–Decoder

The fixed-size summary in our translator

our translator, no attention: Multi30k, 4.0 million parameters, about 9 minutes of training on two processor cores

greedy decoding: **19.5 BLEU** on the 1,000 test captions

source *a girl in karate uniform breaking a stick with a front kick .*

output *une fille en tenue de bain rouge regardant un objet dans un champ .*

“... in a red swimsuit looking at an object in a field”:
fluent, and only its start follows the source



Flickr30k image 101362133

Three Drawbacks of the Recurrent Encoder–Decoder

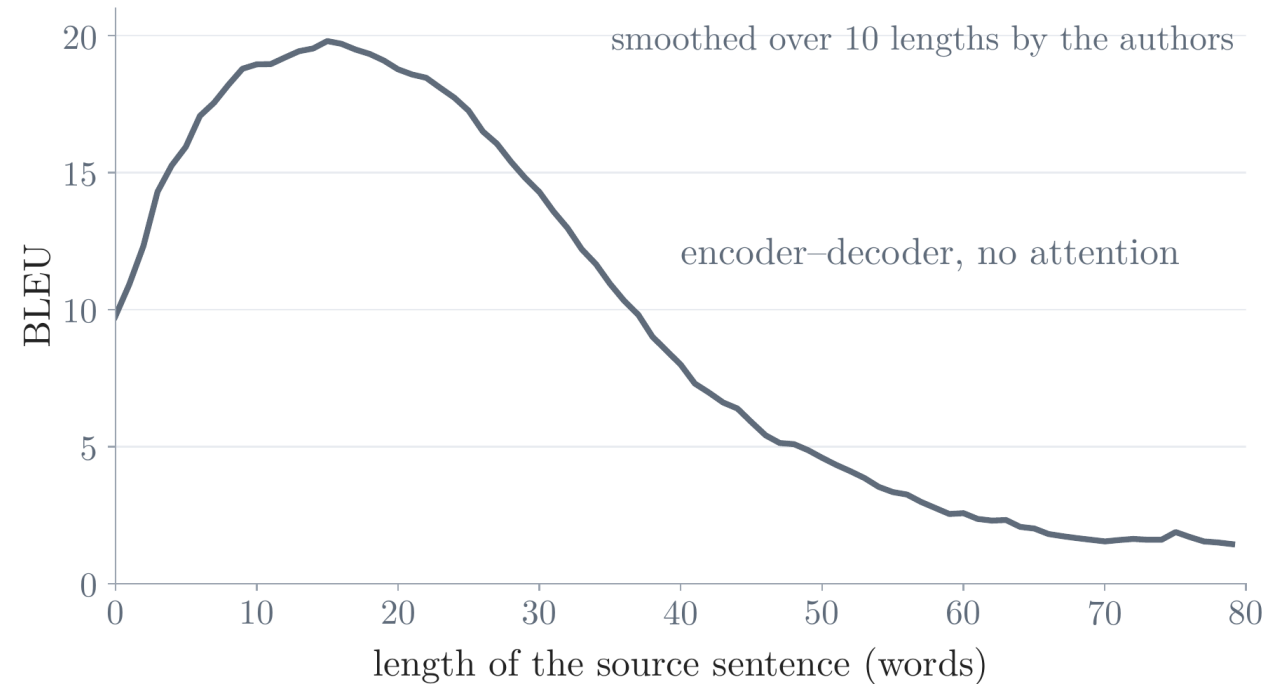
The fixed-size summary in translation

Cho et al. (2014b): an encoder–decoder without attention, trained on English–French

the plot: its BLEU on the test sentences of each length, against the length of the source sentence

highest on short sentences; it “degrades rapidly” as they grow longer

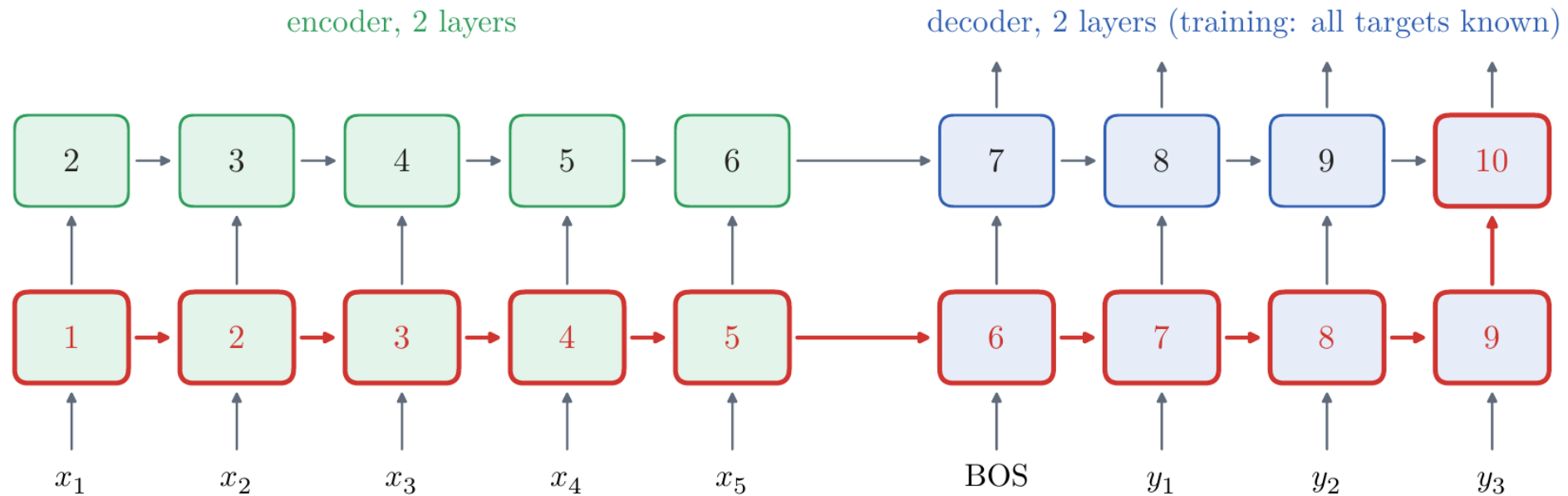
a pressure, not a theorem: Sutskever et al.'s larger model degraded only on the longest



redrawn from Cho et al. (2014b), Fig. 4a (length counted on the source)

Three Drawbacks of the Recurrent Encoder-Decoder

Sequential computation and long paths



each box carries the earliest step at which it can be computed

one position at a time: each state waits for the one before, even in training, when all the inputs are known

long paths: from the first source word to the first prediction, every step in between

$n + r + L - 1$ steps in sequence

n source tokens, r decoder positions, L layers

GNMT (2016): about six days on 96 GPUs

Three Drawbacks of the Recurrent Encoder–Decoder

Familiar remedies, and what fixed the drawbacks

each remedy of 2014 changed **at most one** of the three drawbacks

none gave the decoder more than one summary of the source, or computed a sequence at once

attention: a fresh read of the source at every step (Part 3)

convolution, then self-attention: all positions of a layer at once, in training (Part 4)

	fixed-size summary	sequential computation	long paths
gated cell (LSTM, GRU)	—	—	eased, not shortened
larger state	eased	—	—
bidirectional encoder	—	—	some routes shortened
context at every step	—	—	shortened from c to each prediction
reversed source	—	—	rearranged
attention (Part 3)	removed	—	shortened from source to target
Transformer (Part 4)	removed (cross-attention)	all positions at once, in training	one step between permitted positions

Break

Lecture 3: Sequence Modeling, Translation, and Attention

1. Sequence tasks and translation

sequence modeling; machine translation, how it is learned, decoded, and scored

2. Recurrent encoder-decoders

the encoder-decoder framework; models improved one after another; their drawbacks

3. Attention

its inception and its variants

4. Beyond recurrence

convolutional sequence models, translation from attention alone, self-attention and the Transformer

Attention

The idea

the problem: one summary of fixed size for the whole source (Part 2) → **the fix here:** attention

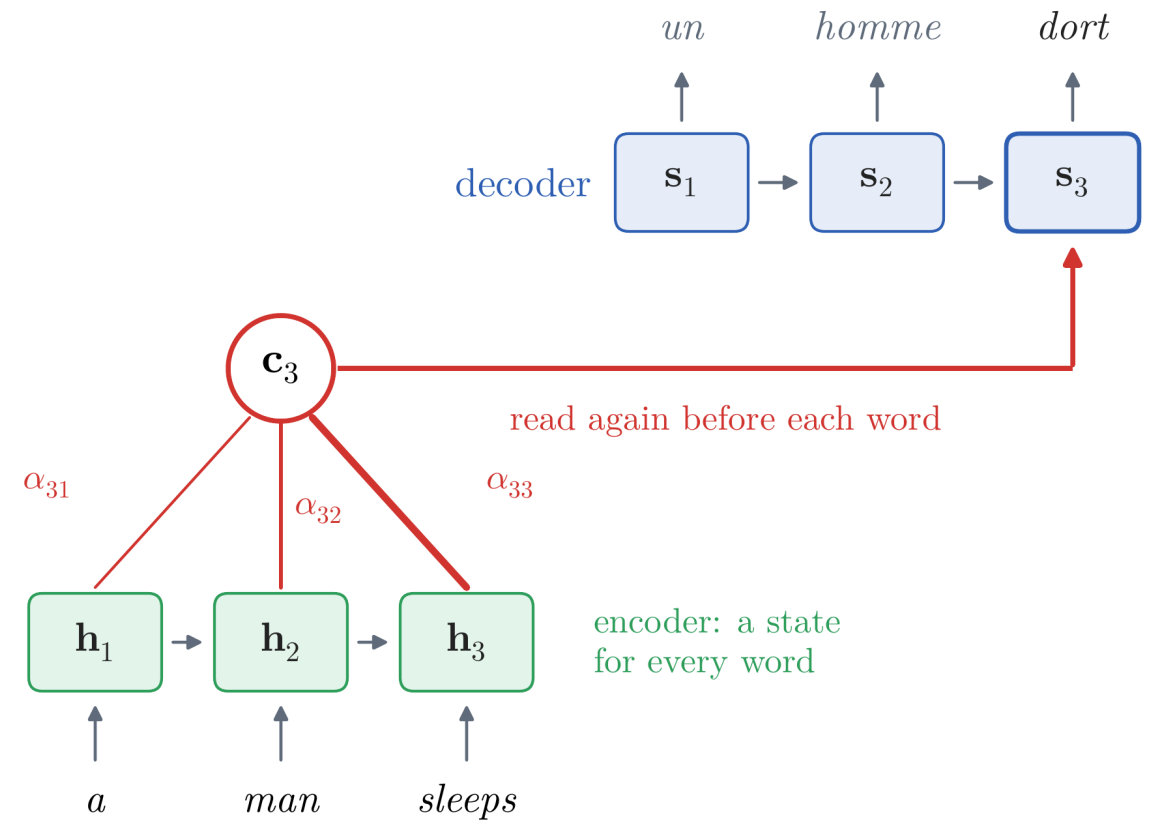
“a fixed-length vector is a bottleneck”

Bahdanau, Cho, and Bengio (2015)

keep a state for every source word (Bahdanau et al.: a bidirectional encoder, each state about its word and the words around it)

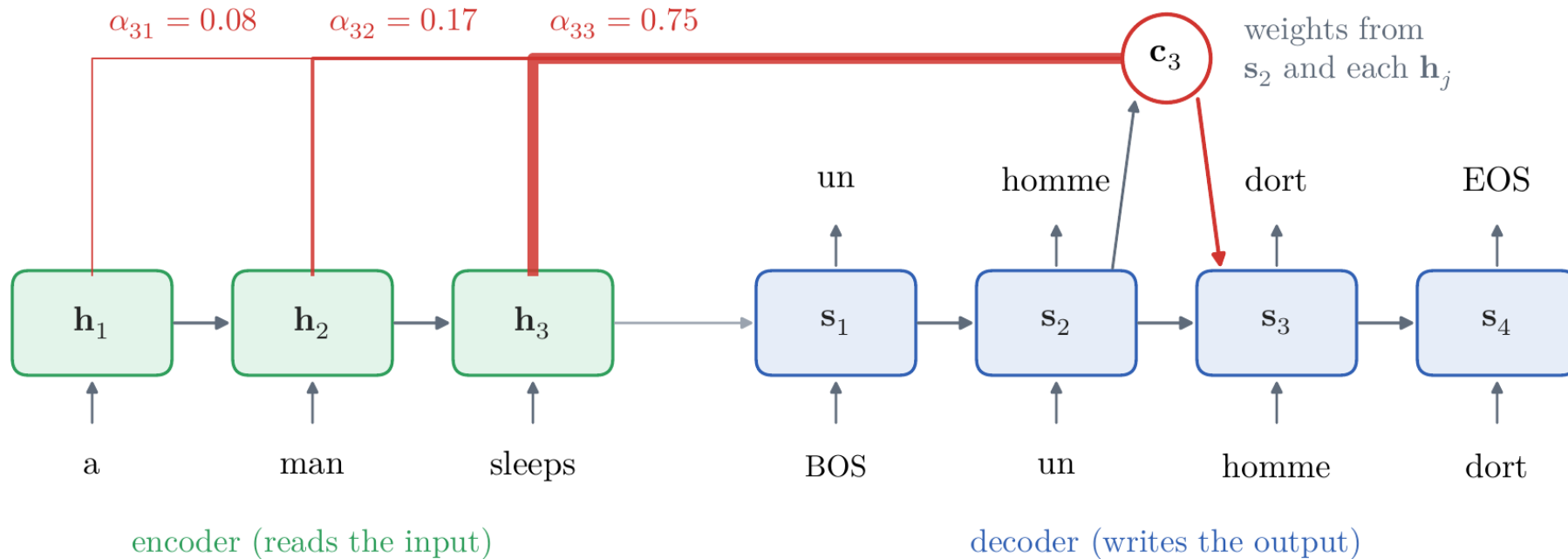
before each word, read them again: a weighted average of the states

the weights: computed from the decoder's state; learned with the rest



Attention

The read



1. score each source state

$$e_{tj} = a(\mathbf{s}_{t-1}, \mathbf{h}_j)$$

2. normalize

$$\alpha_{tj} = \frac{\exp e_{tj}}{\sum_k \exp e_{tk}}$$

3. average

$$\mathbf{c}_t = \sum_j \alpha_{tj} \mathbf{h}_j$$

a new context at every step, from all the source states

Attention

The score, and the context in the decoder

the score: a small network with one hidden layer, trained with the rest (additive attention)

$$a(\mathbf{s}, \mathbf{h}) = \tanh(\mathbf{s}W_a + \mathbf{h}U_a) \mathbf{v}_a^\top$$

the query: the decoder's previous state; it has not read the last word written

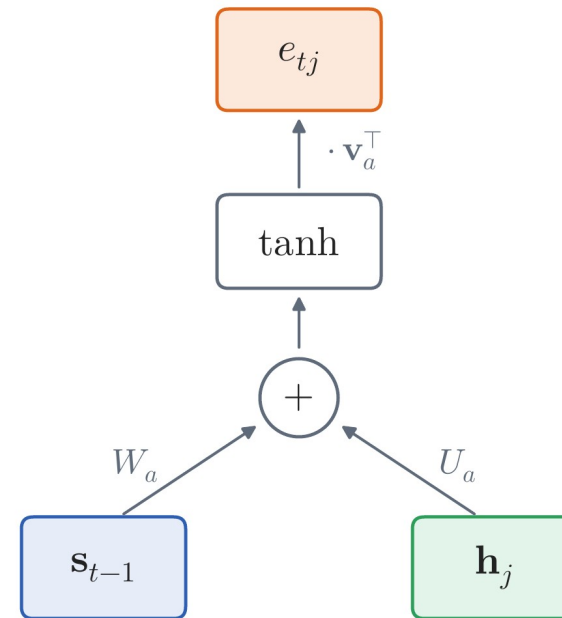
each state: compared with the query, then returned (the variants split the two jobs: a key and a value)

the context enters the decoder's update and its output

$$\mathbf{s}_t = f_{\text{dec}}(\mathbf{s}_{t-1}, E(y_{t-1}), \mathbf{c}_t)$$

$$q(y_t \mid y_{<t}, \mathbf{x}) = \text{softmax}([\mathbf{s}_t; \mathbf{c}_t] U)$$

a set: reorder the source, and the read is the same; order reaches the read only through the states



one hidden layer;
the same network for every j ;
trained with the rest

Attention

Training without alignment labels

nobody labels which word matches which

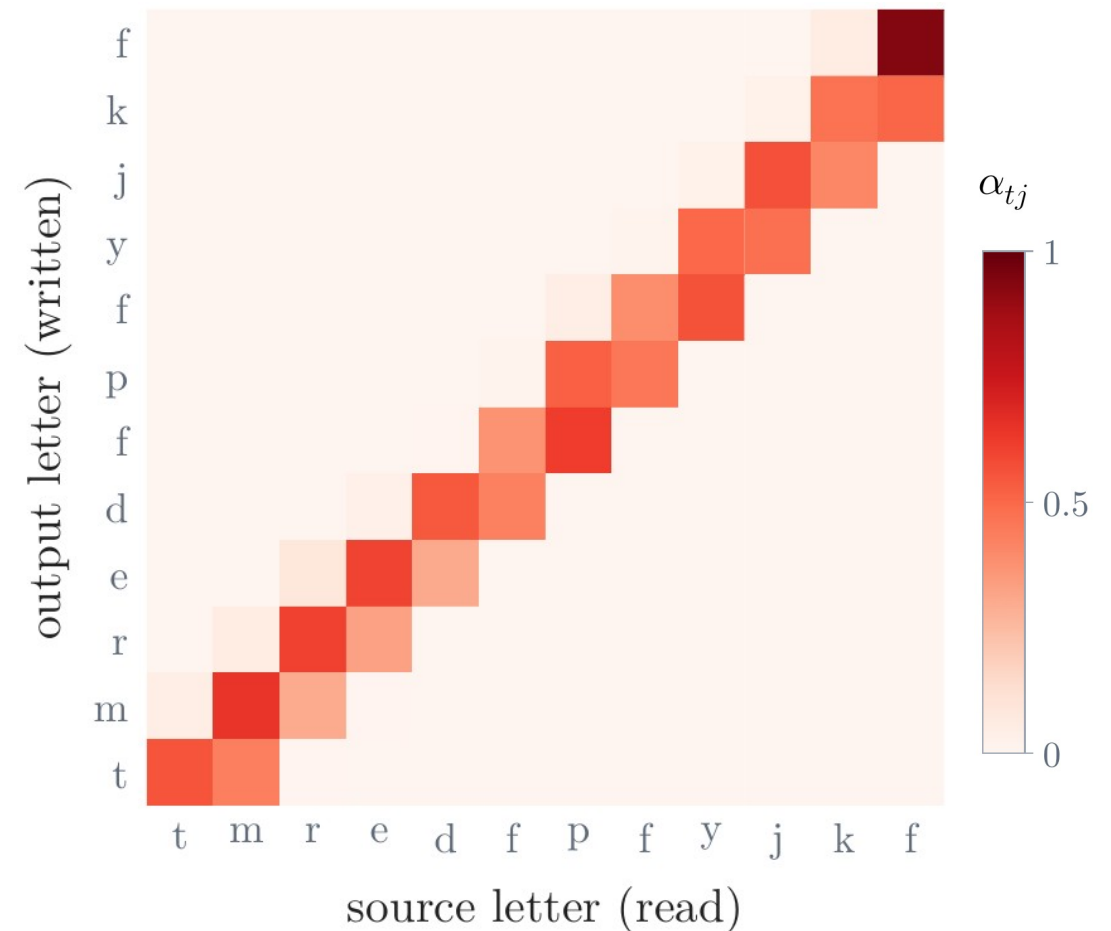
the loss of the next word reaches every score through the context

$$\frac{\partial L}{\partial e_{tj}} = \alpha_{tj} (\mathbf{h}_j - \mathbf{c}_t) \gamma_t^\top, \quad \gamma_t = \frac{\partial L}{\partial \mathbf{c}_t}$$

weight moves between positions; it is never created

the correspondences emerge from the loss alone

the picture: our reversal network with attention, on one 12-letter string; each row sums to 1



each square: the weight on source letter j (column) when writing letter t (row)

Attention

Results: Bahdanau et al. (2015)

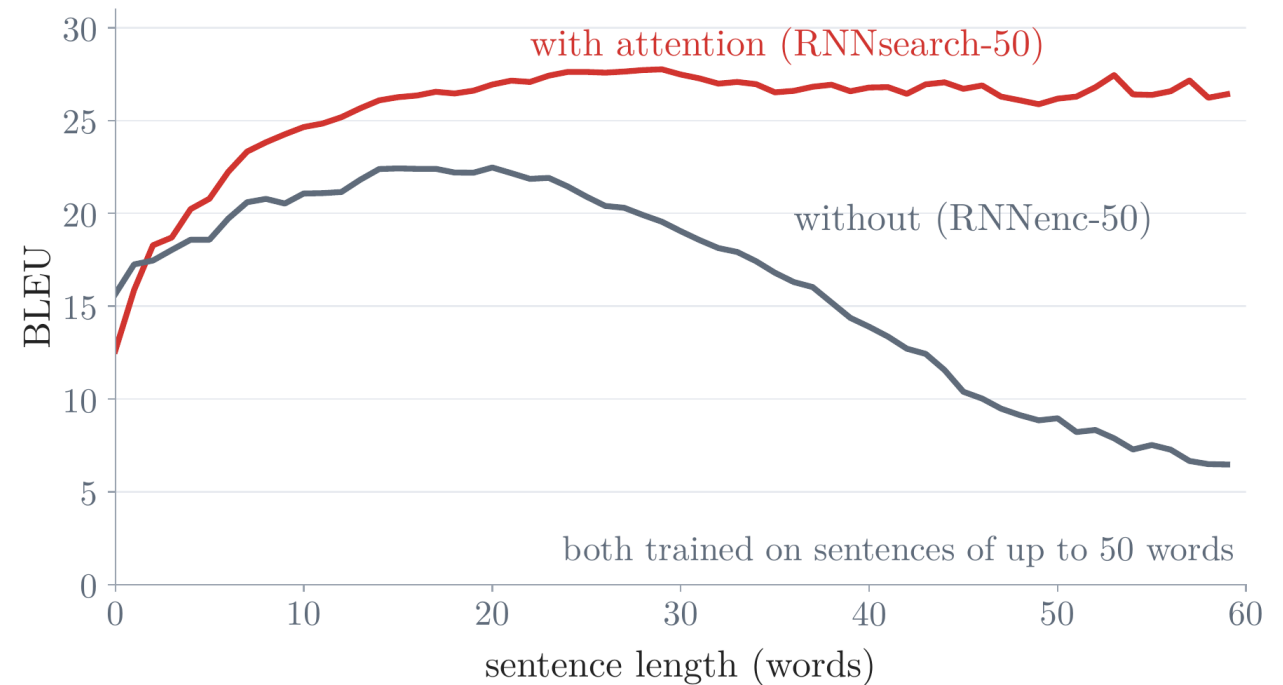
the experiment: WMT 2014 English–French, 30,000-word vocabularies; the same encoder–decoder with and without attention, trained on sentences of up to 30 or 50 words

the plot: BLEU on the test sentences of each length; red with attention, gray without (both trained on sentences of up to 50 words)

without attention, BLEU falls as the sentences grow; with attention it stays level, “even with sentences of length 50 or more”

+8.9 BLEU with attention (26.75 against 17.82)

trained longer, it passed the phrase-based system on sentences without unknown words



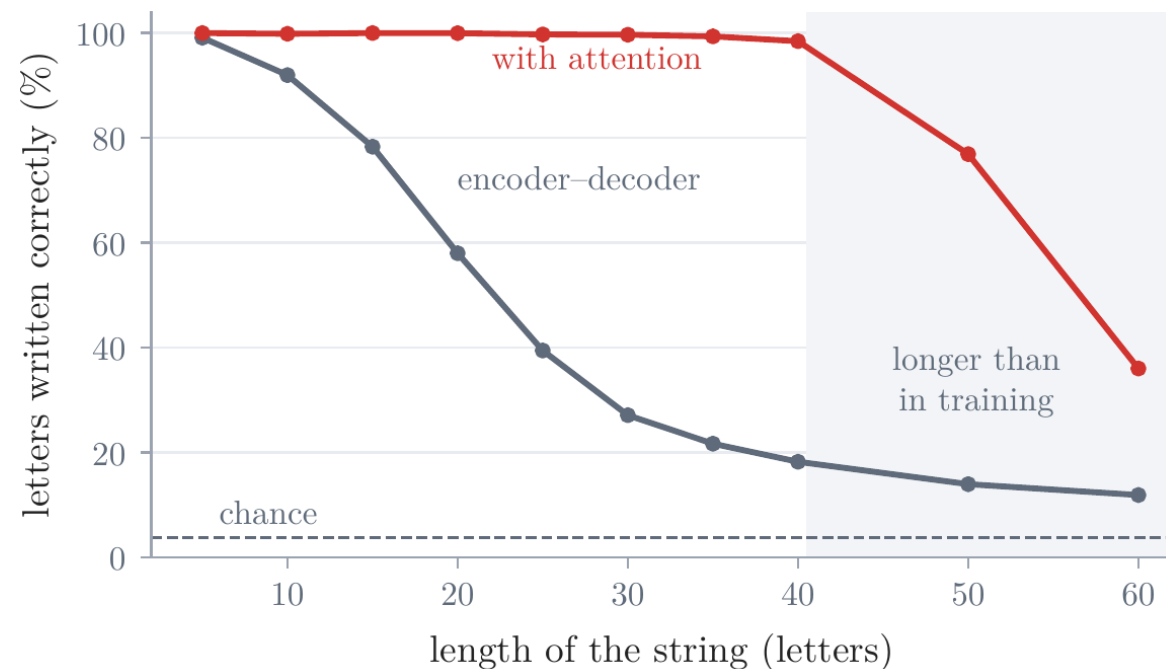
redrawn from Bahdanau et al. (2015), Fig. 2 (RNNsearch-50 and RNNenc-50)

Attention

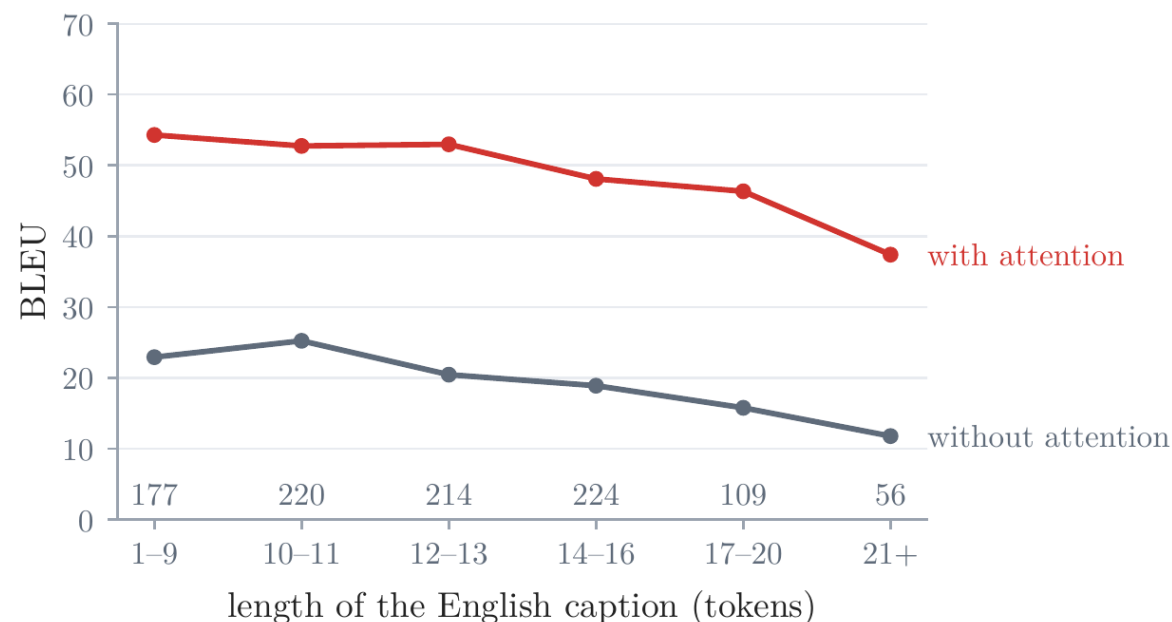
Results: our small models

the experiment: our two models of Part 2, trained again with attention; same data, sizes, steps, and decoding

reversal: over 99% of letters right up to 35, against 18% at 40 without



translation: 49.2 BLEU against 19.5, by caption length



ahead at every length: 37 against 12 even on the 56 longest captions

Attention

Reading the weights: the paper

the picture: RNNsearch-50's weights for every read of one test sentence

each square: the weight α that the read before French word t (its row) gives English word j (its column)

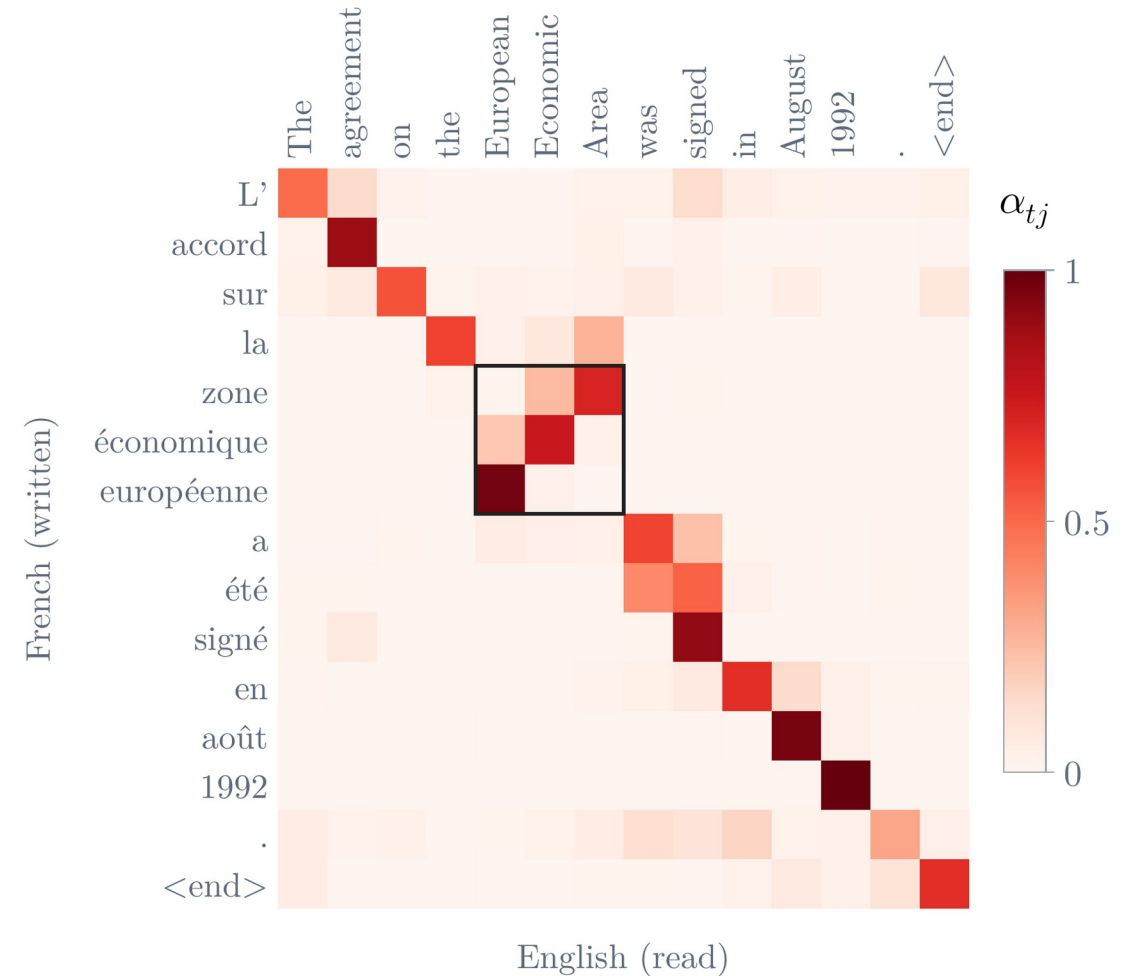
$$\alpha_{tj} = \frac{\exp e_{tj}}{\sum_k \exp e_{tk}}$$

from 0 (white) to 1 (dark red); **each row sums to 1**

mostly diagonal: word for word

European Economic Area → *zone économique européenne*
européenne: the order reversed (outlined)

learned from the translation loss alone



Attention

Reading the weights: where, not why

our translator (the attention model of these slides), decoding the karate caption greedily

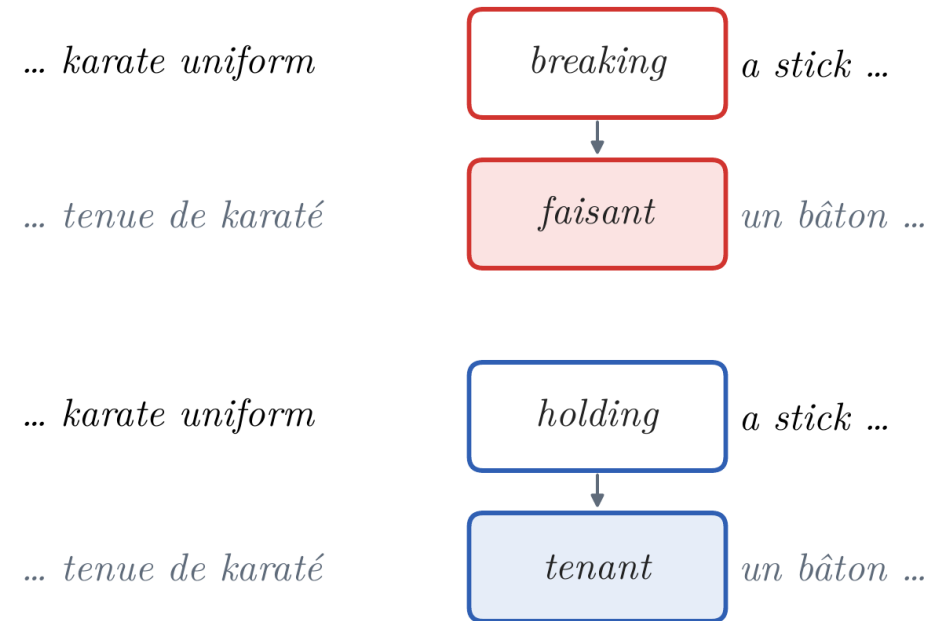
the read is right: at the step that should write *brisant* ('breaking'), 0.73 of the weight is on *breaking*

the choice is not the read alone: the word comes from the context together with the decoder's state (what it has written) and the output layer

here they spoil it: the model is unsure, and its most probable word, *faisant* ('doing'), has probability 0.10

the weights show where the context came from, not why the word was chosen

causes need interventions: *holding* for *breaking* gives *tenant*, and nothing else changes



the model's translations, with one source word changed

Attention Variants

Queries, keys, and values

query: what the reader is looking for

key: what an item is found by

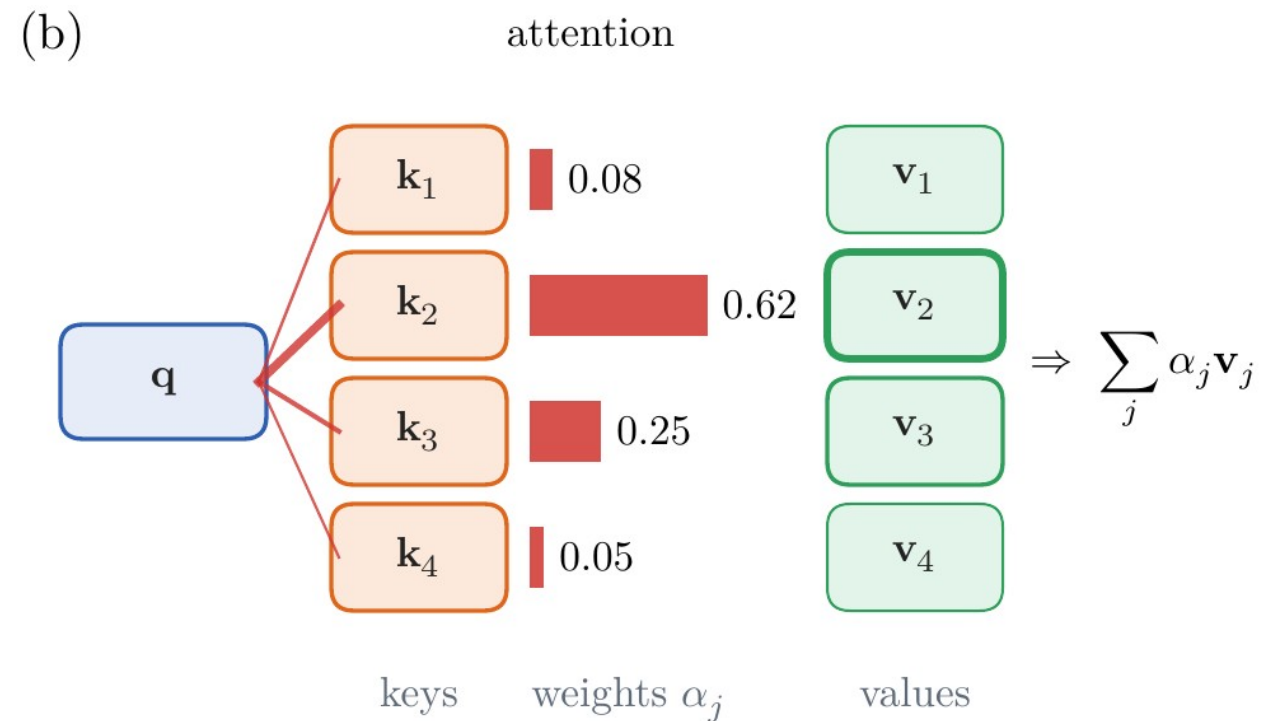
value: what the item returns when read

a dictionary: returns the value of the one key that matches the query

attention: compares the query with every key, turns the comparisons into weights, and returns the weighted average of the values

in Bahdanau et al., the key and the value of a position are the same **h**; they need not be (Part 4)

the words come from memory networks: the Neural Turing Machine (2014) looked up a row of its memory with a “key vector”; key-value memory networks (2016)



Attention Variants

How to score

dot product: no parameters; the two sizes must match

$$\mathbf{s} \mathbf{h}_j^\top$$

bilinear: a dot product after learned maps

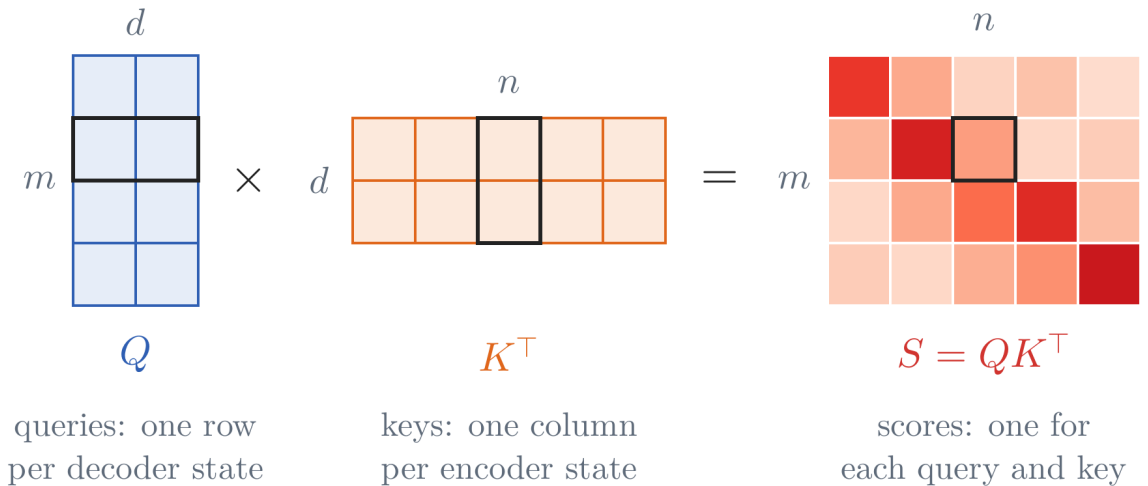
$$\mathbf{s} W \mathbf{h}_j^\top = (\mathbf{s} W_Q)(\mathbf{h}_j W_K)^\top$$

additive: a small network (Bahdanau et al.)

$$\tanh(\mathbf{s} W_a + \mathbf{h}_j U_a) \mathbf{v}_a^\top$$

all queries against all keys: one matrix product, what parallel hardware does fastest (dot and bilinear)

Lecture 4: the bilinear score, factored and scaled



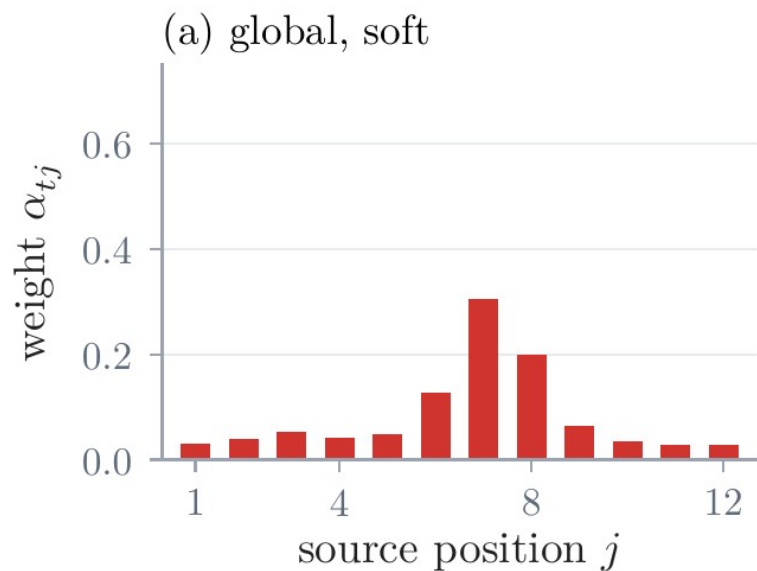
every query against every key, in one matrix product

Attention Variants

Where and how to read

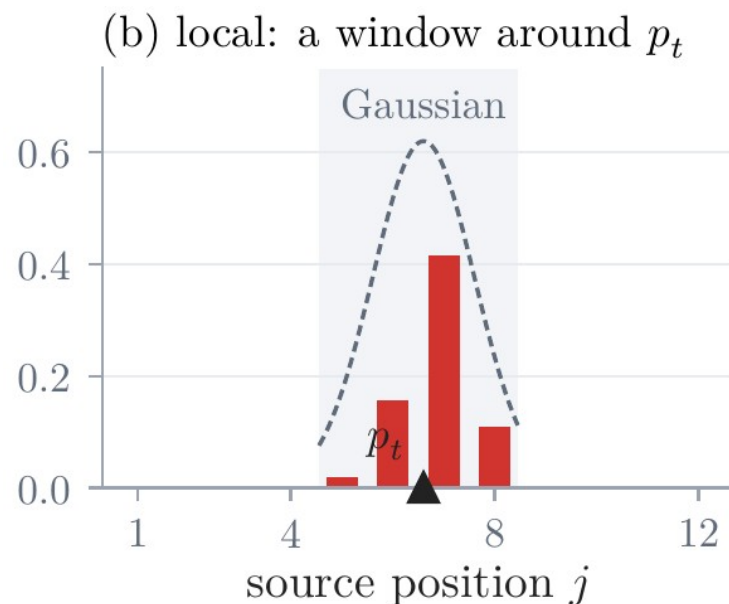
global, soft

every position; the average under the weights



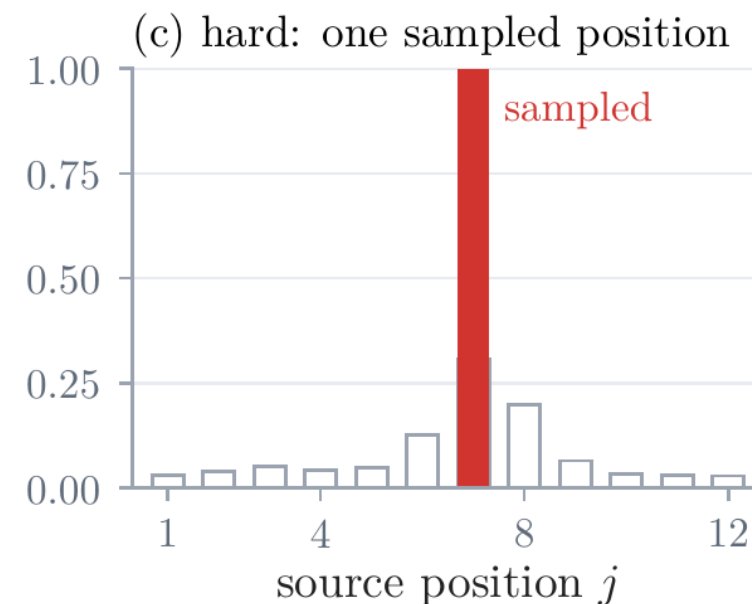
local

a window around a predicted center



hard

one sampled position; no derivative



soft and global became the standard: plain backpropagation

Attention Variants

When to read

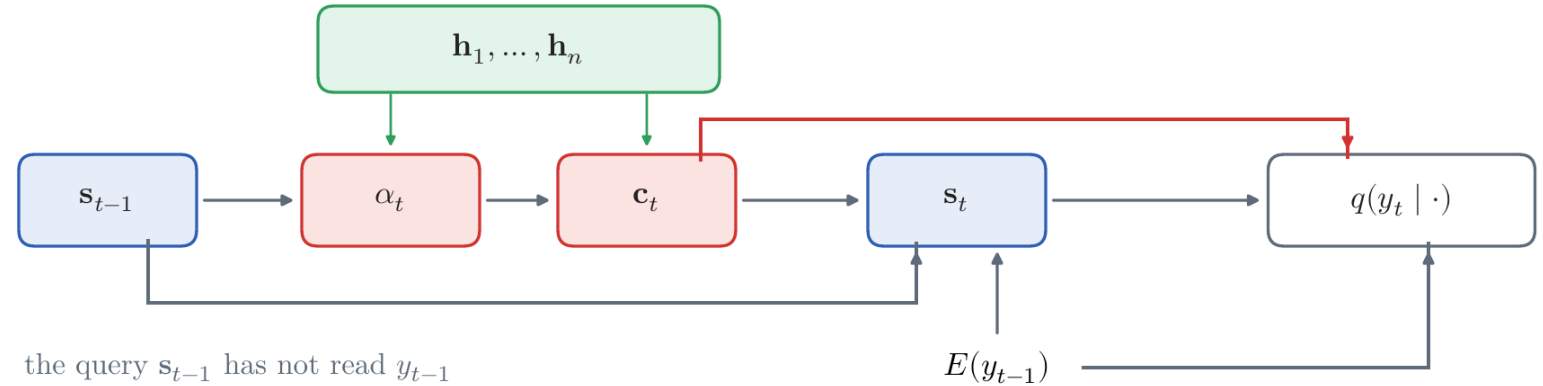
Bahdanau: read with the previous state, then update

Luong: update, then read with the new state

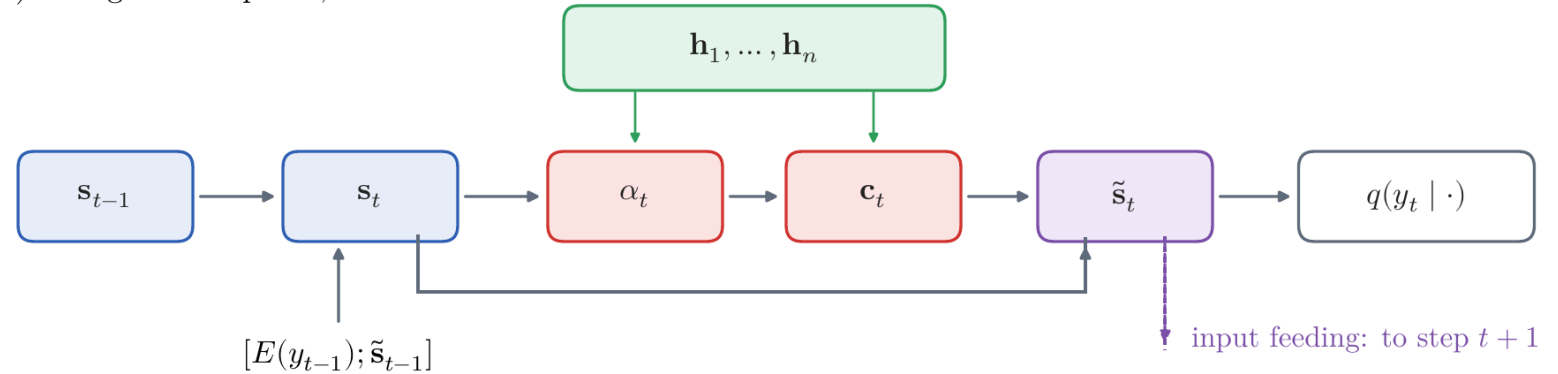
input feeding: the read joins the next step's input

the same read, placed differently

(a) Bahdanau et al.: read, then update



(b) Luong et al.: update, then read



Attention Variants

Attention beyond translation: images

a **caption generator**: image regions are the source; the decoder reads them before each word



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Xu et al. (2015), Fig. 3: where the model read (white) when it wrote the underlined word

the same read, over image regions instead of source words

the read is everywhere; the states it reads still come one after another

Attention

Attention at scale: Google's translator (2016)

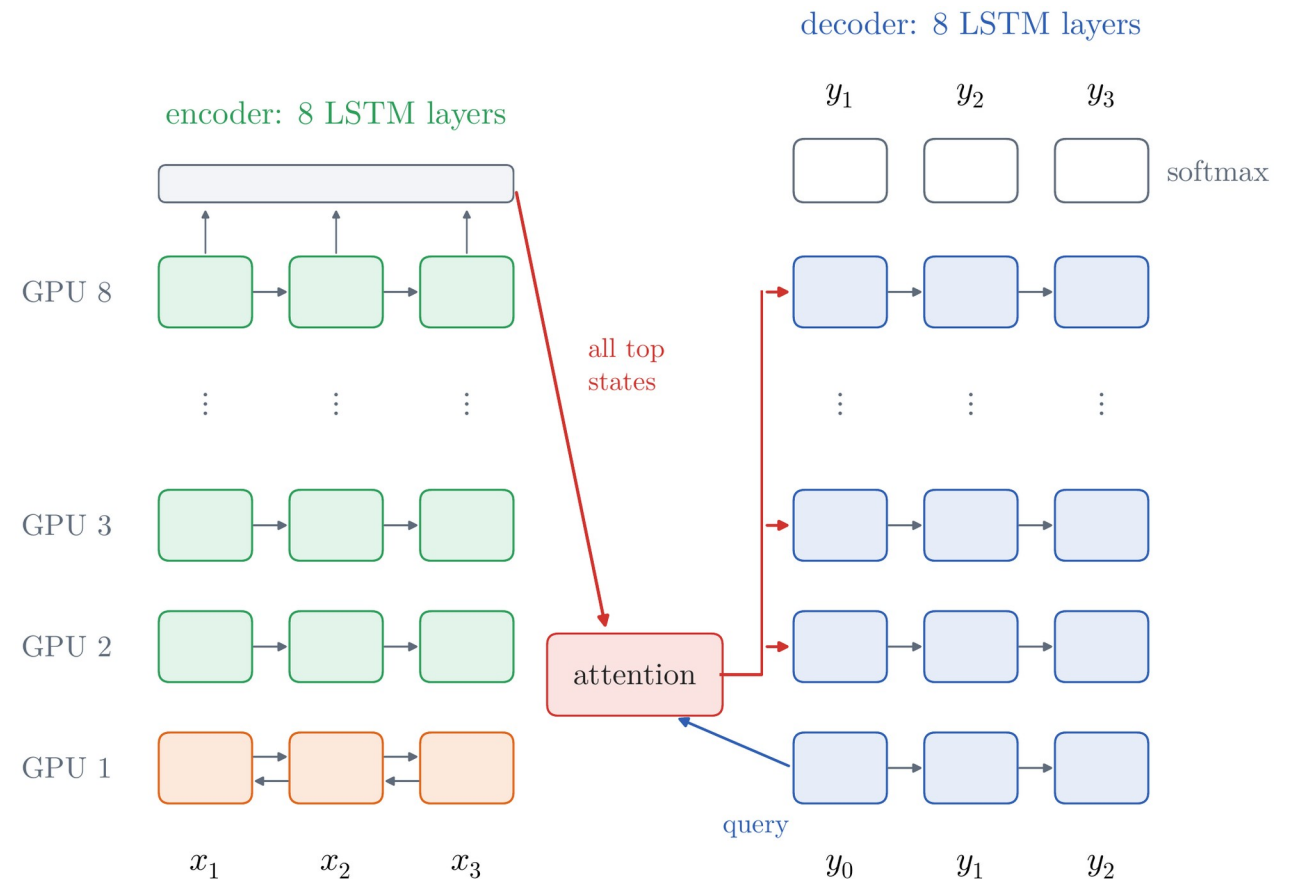
GNMT (Wu et al., 2016): the attention model, made large

8 LSTM layers in the encoder, 8 in the decoder, with residual connections; word pieces, so no unknown words

one model: 39.0 BLEU English–French

in production from September 2016, replacing the phrase-based system for Chinese–English

but slow to train: about six days on 96 GPUs for one model



redrawn and simplified from Wu et al. (2016), Fig. 1

Lecture 3: Sequence Modeling, Translation, and Attention

1. Sequence tasks and translation

sequence modeling; machine translation, how it is learned, decoded, and scored

2. Recurrent encoder-decoders

the encoder-decoder framework; models improved one after another; their drawbacks

3. Attention

its inception and its variants

4. Beyond recurrence

convolutional sequence models, translation from attention alone, self-attention and the Transformer

Convolutional Sequence Models

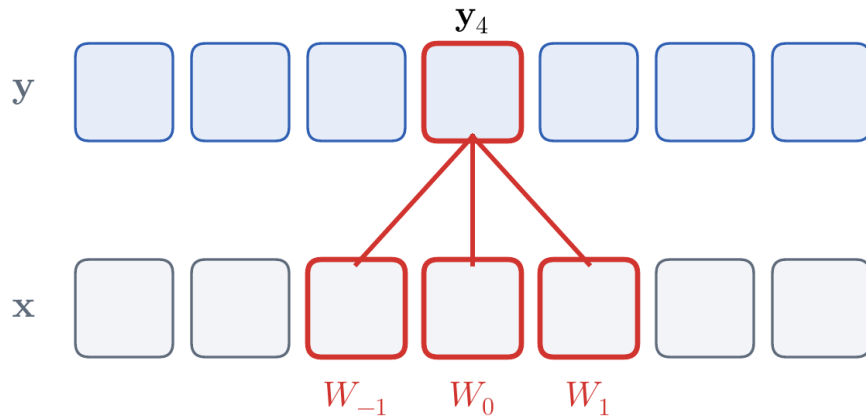
Convolution over a sequence

the problem: one position at a time, even in training (Part 2) → **the fix here:** convolution

each state from a small window of the layer below, not from the state before

$$\mathbf{y}_i = \mathbf{b} + \sum_{u=-(k-1)/2}^{(k-1)/2} \mathbf{x}_{i+u} W_u$$

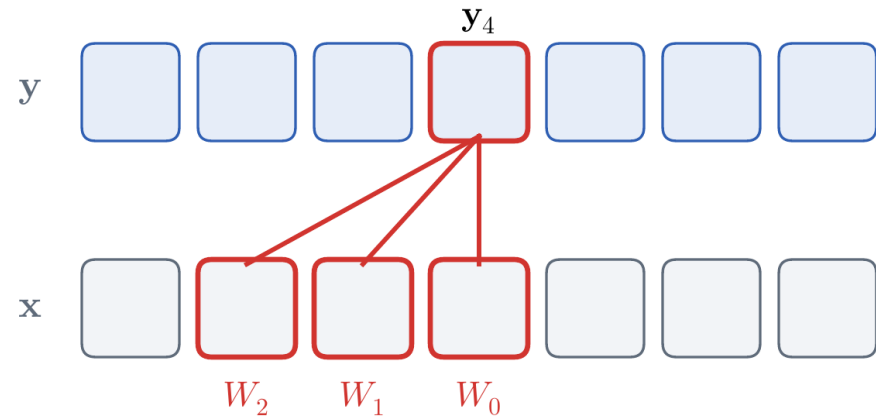
centered, for an encoder: both sides



the same weights at every position

$$\mathbf{y}_t = \mathbf{b} + \sum_{u=0}^{k-1} \mathbf{x}_{t-u} W_u$$

causal, for a decoder: itself and the past



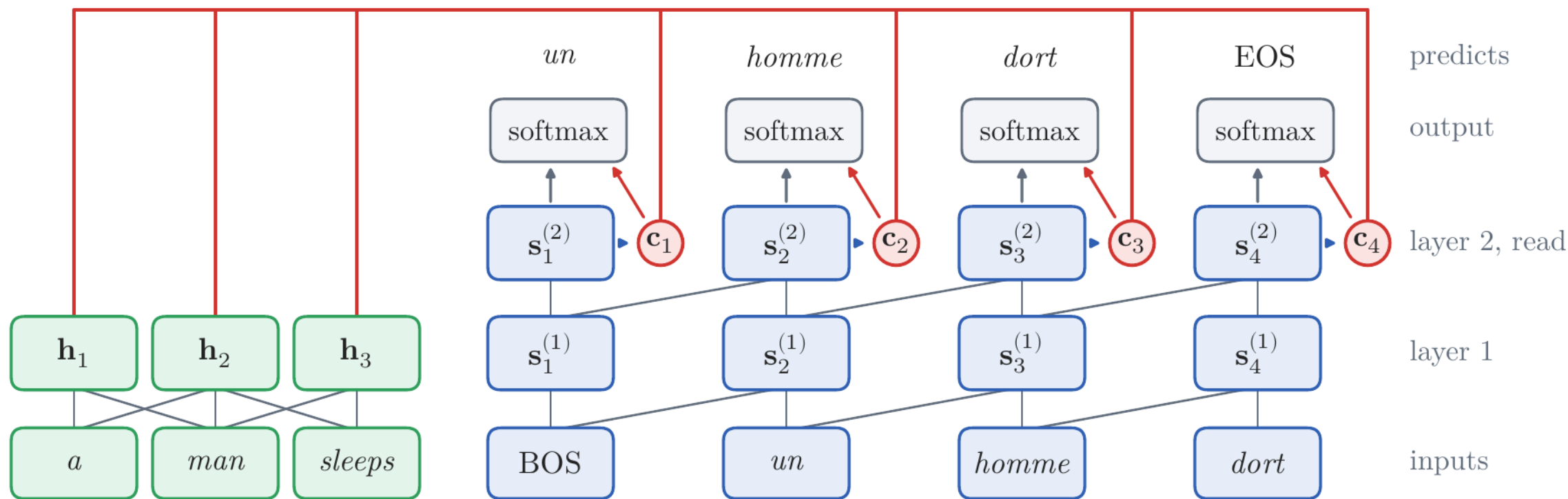
no dependence along the layer: all outputs at once

Convolutional Sequence Models

A small convolutional translator: training

(a) Training: each layer computes all its positions at once

h_1, h_2, h_3 : keys and values of every read



encoder: one centered convolution; all source states at once

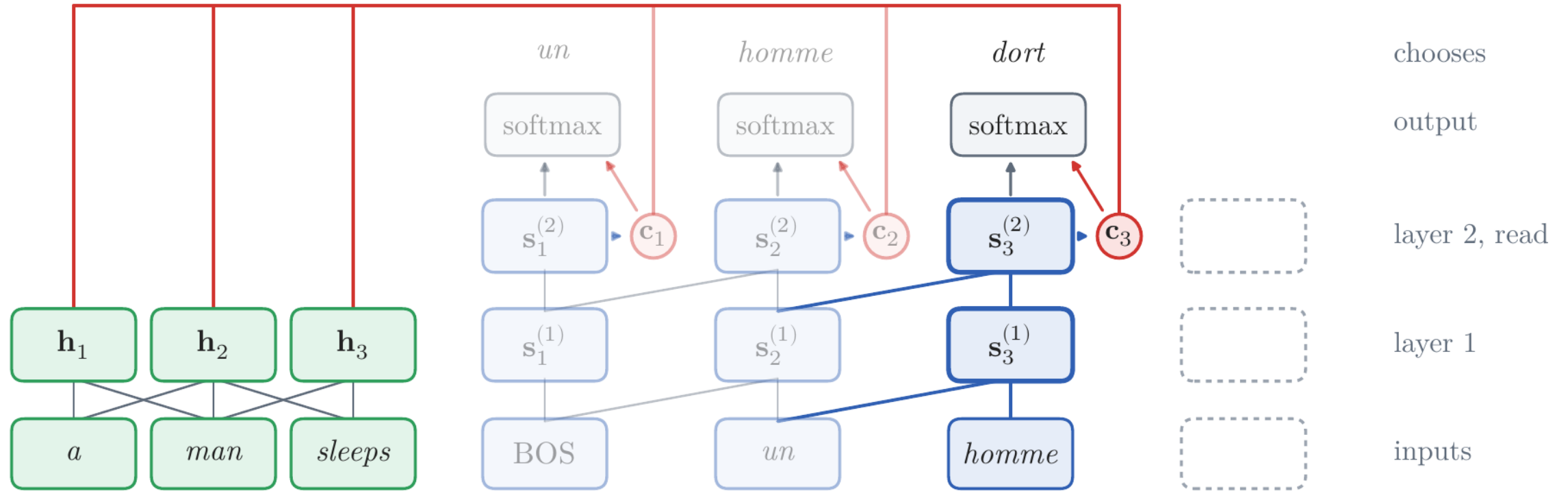
decoder: two causal convolutions, then a read of the source and a softmax, at all positions at once

Convolutional Sequence Models

A small convolutional translator: generation

(b) Generation, step 3: only the new column is computed

h_1, h_2, h_3 : keys and values of every read



one new column per word; the earlier columns are kept

one word at a time, as for every model: each input is the word chosen before

Convolutional Sequence Models

How far a convolution reads

L layers of width k reach

$$1 + (k - 1)L$$

with dilations 1, 2, 4, ...

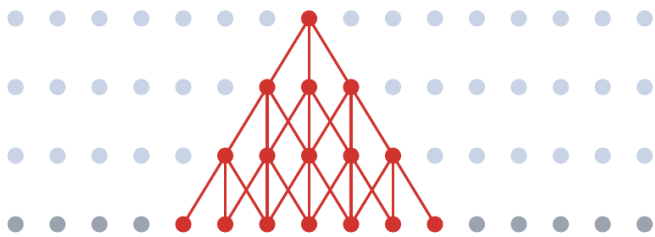
$$1 + (k - 1)(2^L - 1)$$

distance costs layers

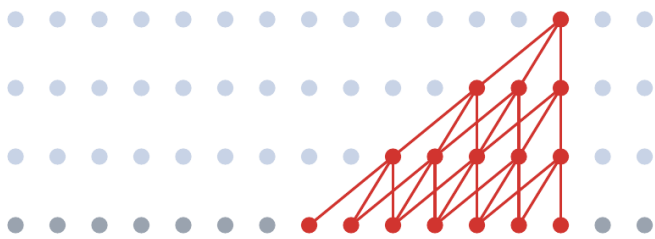
position 16 from position 1, width 3: 15
centered layers, 8 causal

attention: one step

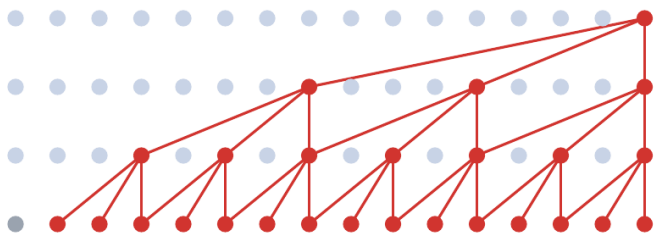
(a) centered, width 3



(b) causal, width 3

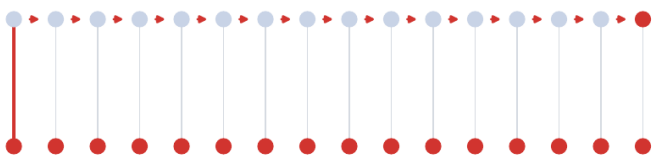


(c) causal, width 3, dilations 1, 2, 4



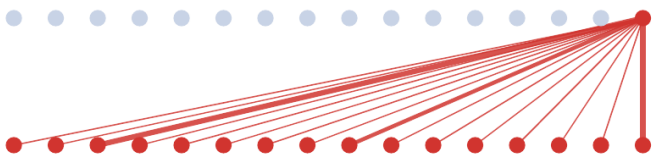
(d) recurrent layer

the first input reaches the last output through 15 steps



(e) causal attention layer

every earlier input in one step, weighted by content



(f) For output 16 to depend on input 1:

centered, width 3	15 layers
causal, width 3	8 layers
causal, dilations 1, 2, 4, 8	4 layers
recurrence	15 steps in sequence
attention	1 layer

Convolutional Sequence Models

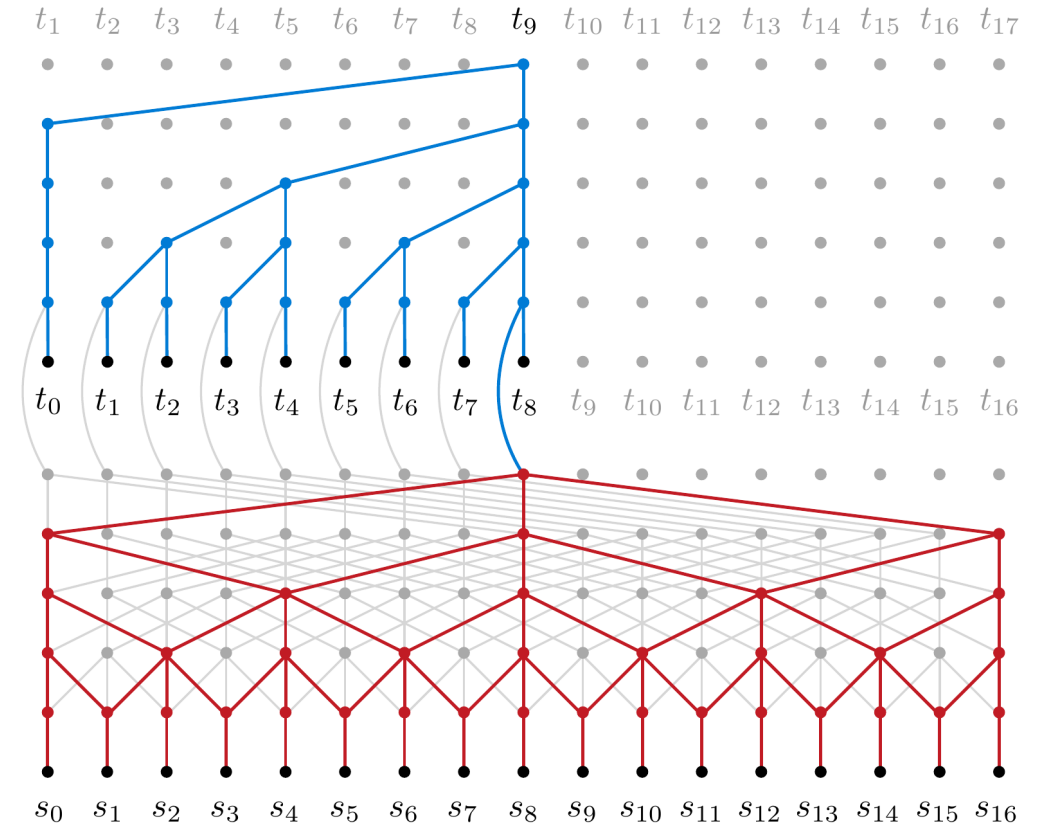
ByteNet: convolution only

dilated convolutions, no recurrence, no attention

the decoder sits on the encoder, position by position

access to the source by position, not by content

23.75 BLEU, English-German, character by character



Kalchbrenner et al. (2016), Fig. 1: ByteNet

Convolutional Sequence Models

ConvS2S

ConvS2S (Gehring et al., 2017): the small translator's design, with three changes

1. **gated blocks**: a convolution, a gate on each number, and a residual connection

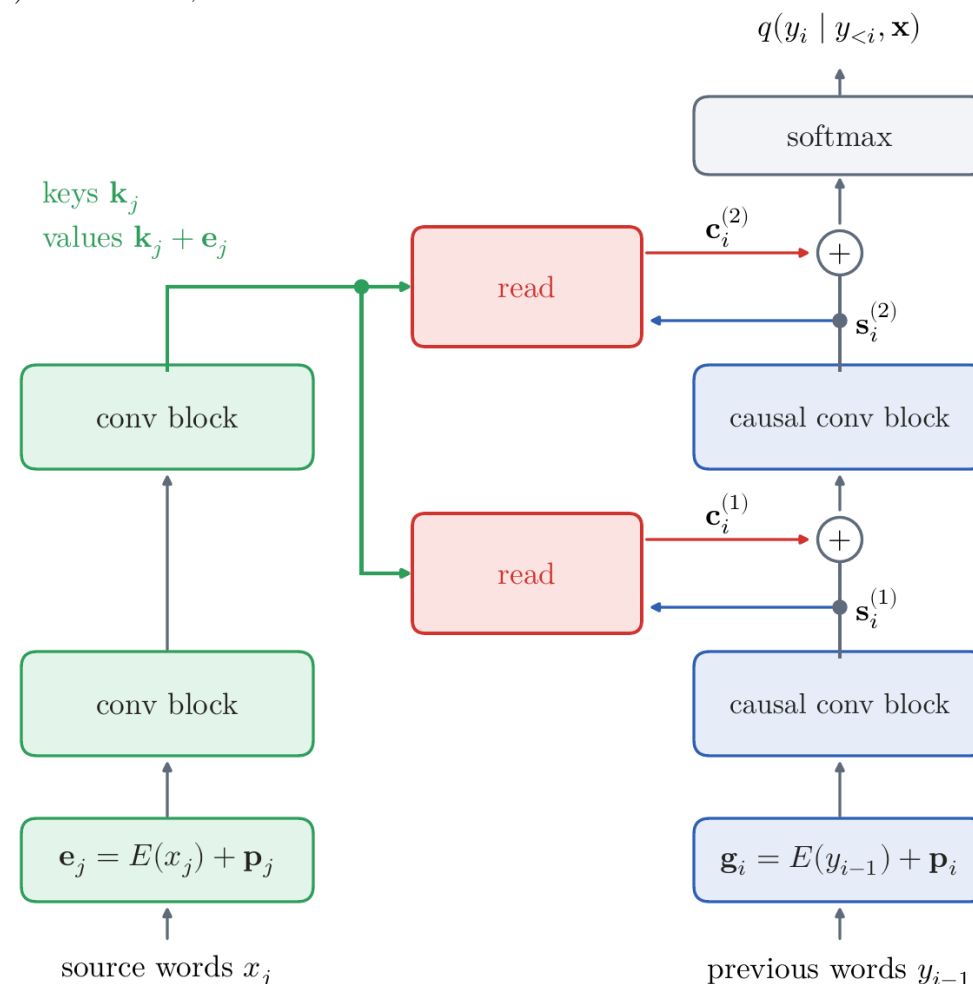
2. **position vectors** added to the word vectors

3. **a read in every decoder layer**: keys $\mathbf{k}$, values $\mathbf{k} + \mathbf{e}$, what was computed and which word

WMT 2014, single models: **40.51 BLEU** English–French and 25.16 English–German, above Google's recurrent 38.95 and 24.61; “an order of magnitude faster”

the limit: distant positions meet only through many layers

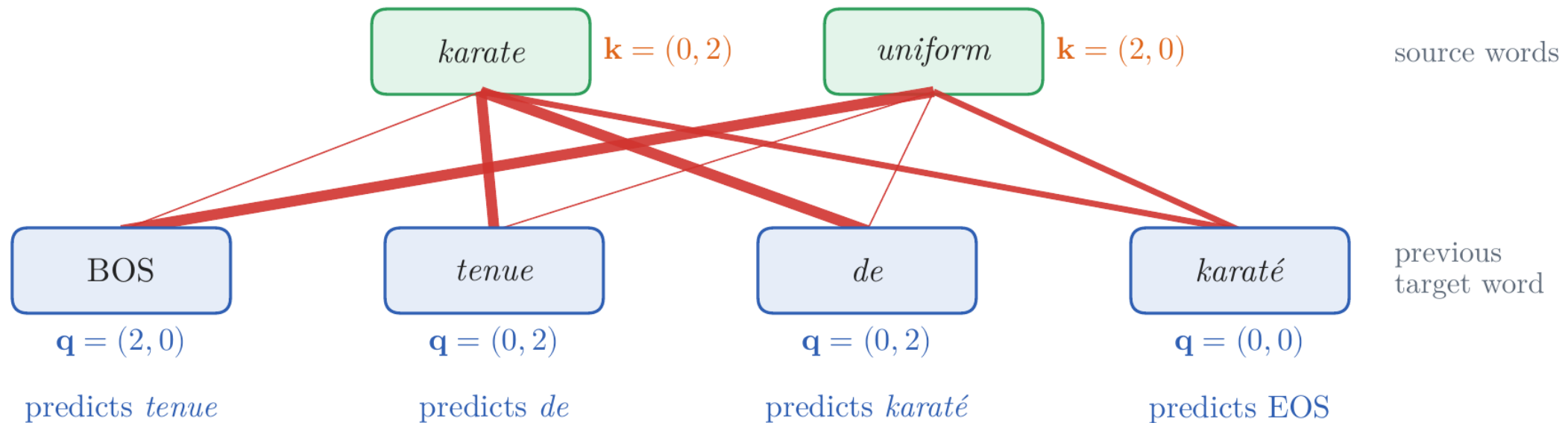
(a) The model, two blocks in each stack



Translation from **Attention Alone**

The model

1. **query** from the previous target word
2. **read** the source: dot products, softmax, average
3. **predict** from the read and the previous word



no encoder computation, no decoder state: **it reorders by content**

Translation from **Attention Alone**

What it lacks

in training: **a few matrix products**; no position waits for another

$$S = QK^\top, \quad A = \text{softmax}(S), \quad C = AV$$

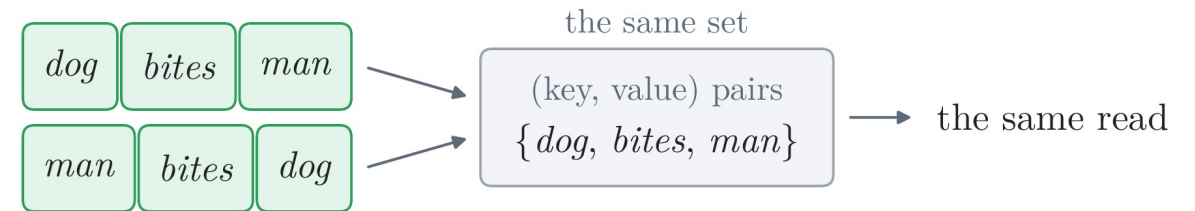
word order: *dog bites man* reads like *man bites dog*

the context of a source word: the same *bank*, by the river or raising rates

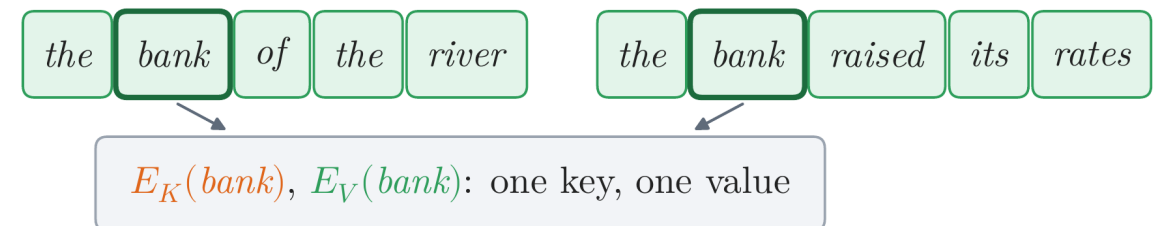
the target so far: after each *un*, the same read

the answers: positions; self-attention in the encoder; causal self-attention in the decoder

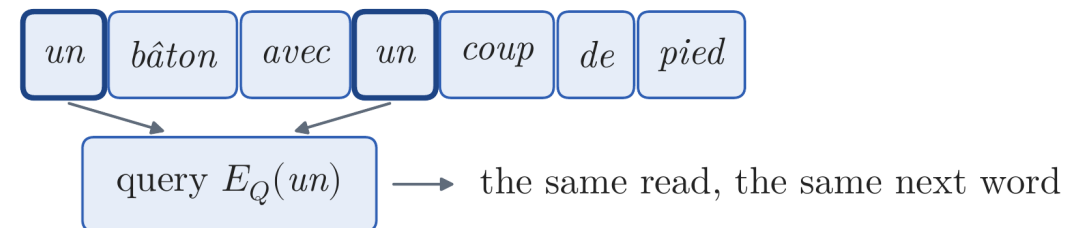
word order



the context of a source word



the target so far



Self-Attention and the Transformer

Self-attention

a sequence reads itself

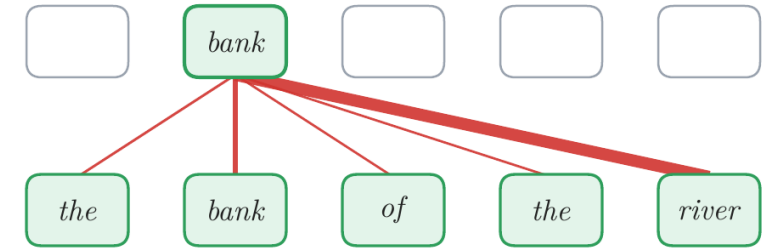
encoder: every position reads every position

decoder: causal, itself and the positions before

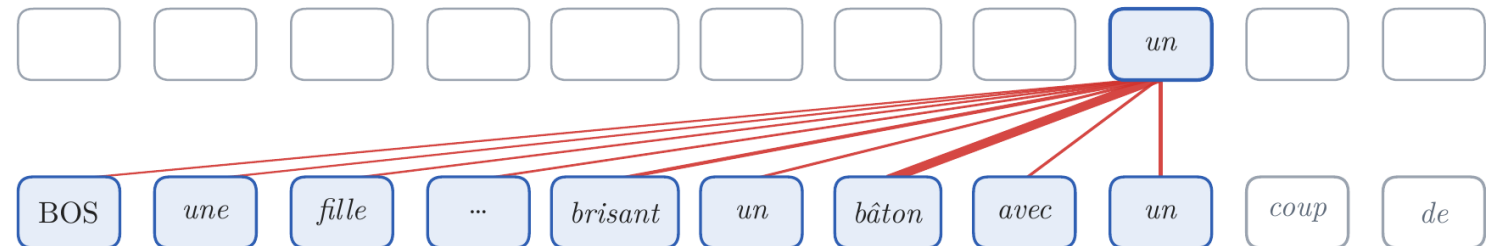
$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{v}_j$$

$$\alpha_{ij} = \text{softmax}_j(\text{score}(\mathbf{q}_i, \mathbf{k}_j))$$

(a) encoder self-attention: every position may read every position



(b) causal decoder self-attention: a position reads itself and the positions before it



all positions at once, like convolution; any position in one step, like cross-attention

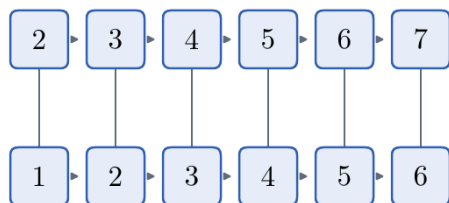
Self-Attention and the Transformer

Recurrence, convolution, self-attention

each box: the earliest step at which it can be computed, with one processor per state

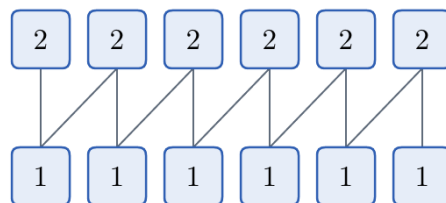
recurrence

in training, all inputs are known



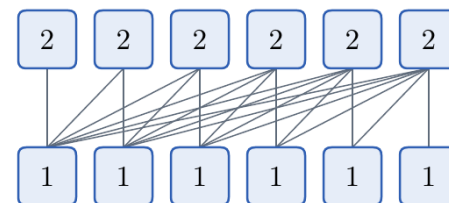
7 steps

causal convolution, width 2



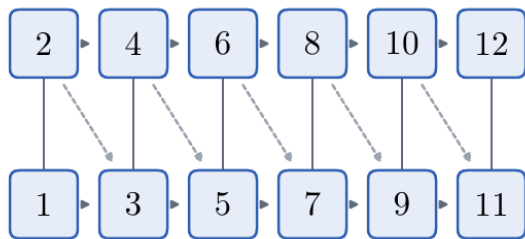
2 steps

causal self-attention

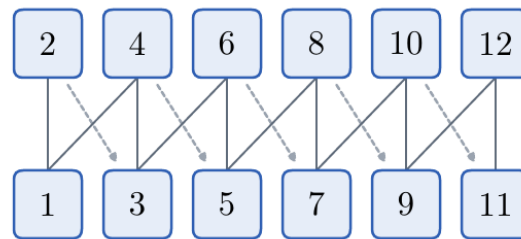


2 steps

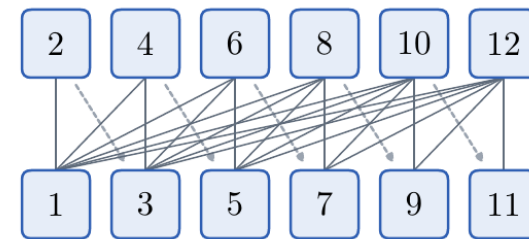
in generation, each input is the previous output



12 steps



12 steps



12 steps

in training: 7 steps, 2, and 2

in generation: one position at a time, for all three

the price: pairwise work, growing with the square of the length

Self-Attention and the Transformer

The original Transformer

encoder self-attention: each source word in its context

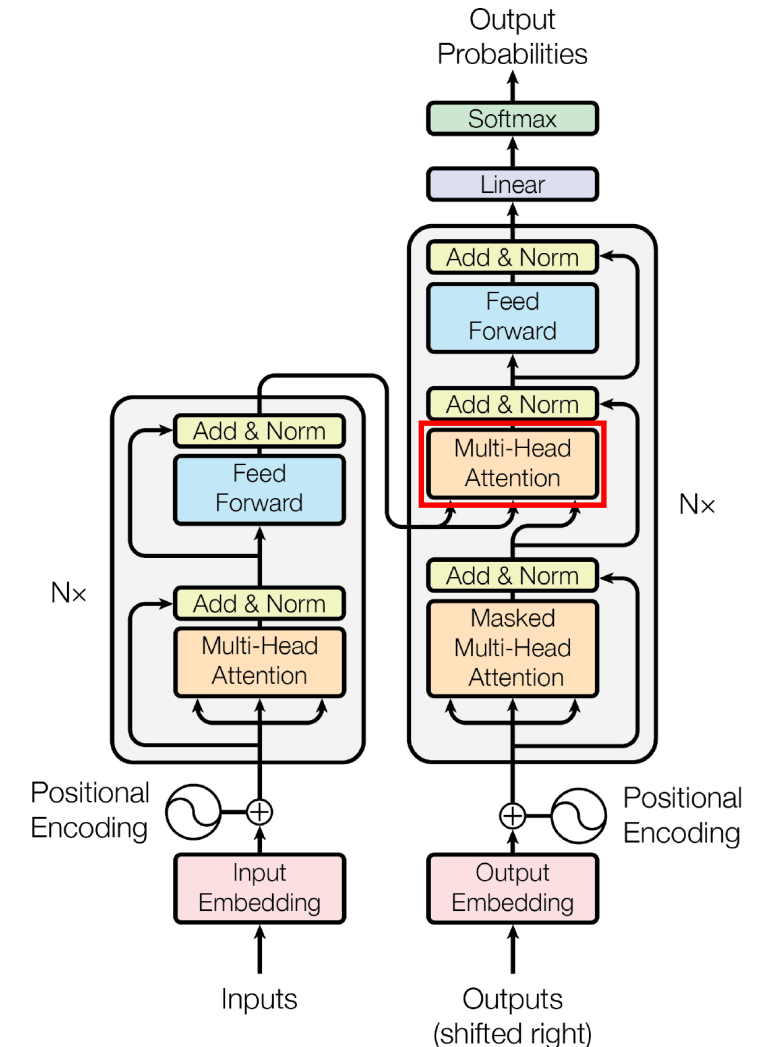
causal decoder self-attention: what has been written

cross-attention: the decoder reads the encoder

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

positions added to the word vectors; several heads; six layers each

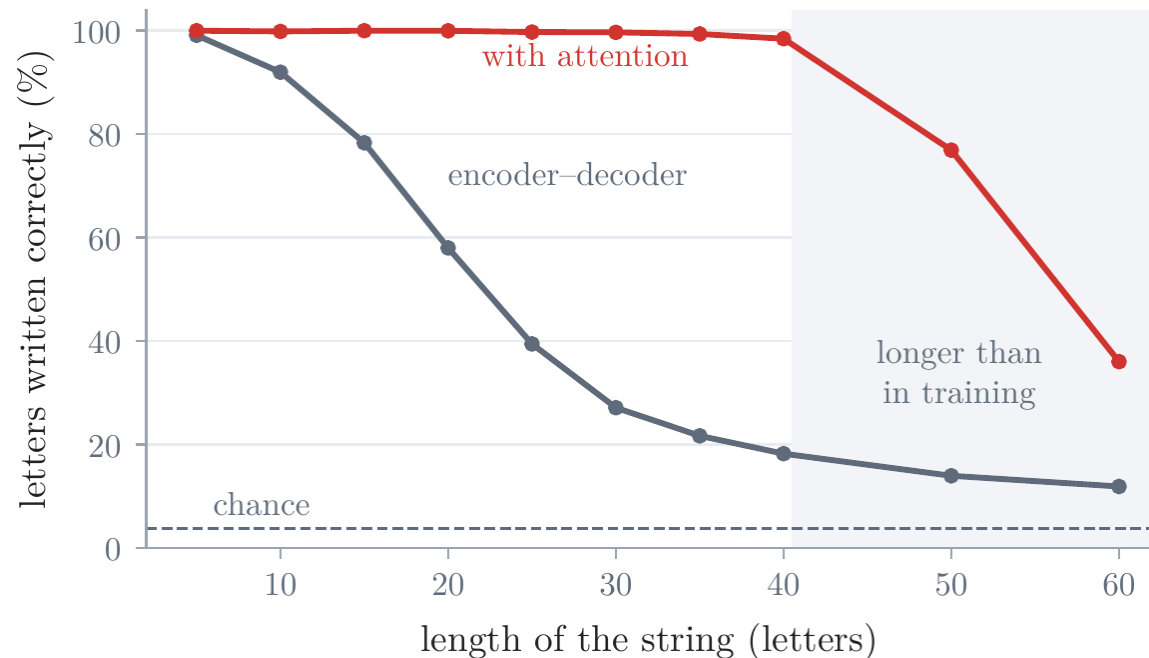
Lecture 4 builds it



Vaswani et al. (2017), Fig. 1

What Carries Forward

Two problems, two answers



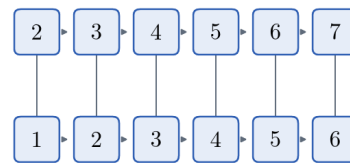
access to the source → attention

a fresh read at every step, learned from the loss

next: the Transformer, built and trained

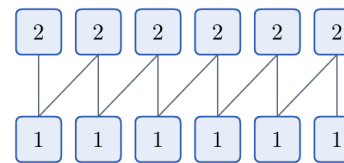
recurrence

in training, all inputs are known



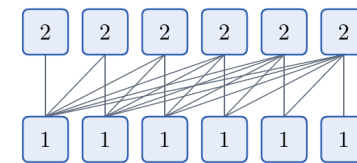
7 steps

causal convolution, width 2



2 steps

causal self-attention



2 steps

one position at a time → self-attention

all positions of a layer at once; any permitted pair in one step