

DDA4220 · Fall 2026

Lecture 11

Deep Learning and Applications

Sequence-to-sequence models and attention

Zhen Liu

CUHK-Shenzhen

Lecture outline

| 01 Sequence-to-sequence models

02 Attention

03 Forecasting and image captioning

04 Self-attention

* denotes optional slides

Conditional sequence prediction

Language modeling

The red car $\longrightarrow$ stopped.

Conditional generation

The red car stopped. $\longrightarrow$ La voiture rouge
s'est arrêtée.

Source sentence • English

Target sentence • French

Source and target sequences

Source • English

$x_1, \dots, x_S$

The

red

car

stopped.

Length and word order
can change

Target • French

$y_1, \dots, y_T$

La

voiture

rouge

s'est

arrêtée.

La = the voiture = car rouge = red

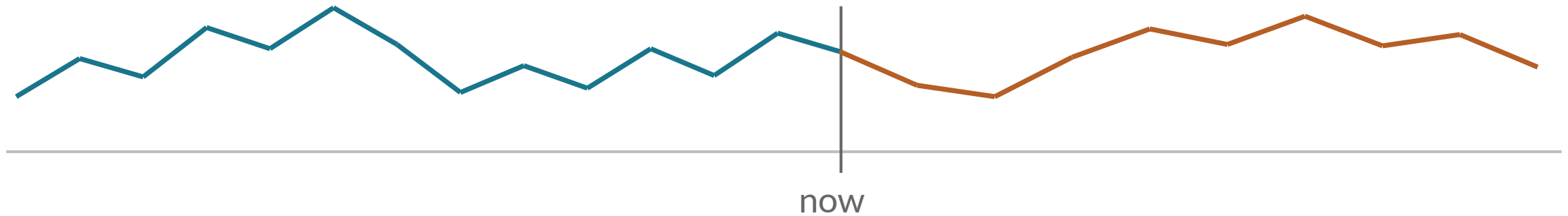
S, T: sequence lengths

Beyond translation

Forecasting

Observed measurements

Future measurements



Speech recognition



The red car stopped.

Autoregressive conditional prediction

Autoregressive: predict each target token from earlier targets and the source

$$p_{\theta}(y_1, \dots, y_T \mid x_1, \dots, x_S) = \prod_{t=1}^T p_{\theta}(y_t \mid y_{<t}, x_{1:S})$$

$x = x_{1:S}$

Source sequence

$y_{<t}$

Previous targets

First predictions

$$p(\text{La} \mid [\text{BOS}], x)$$

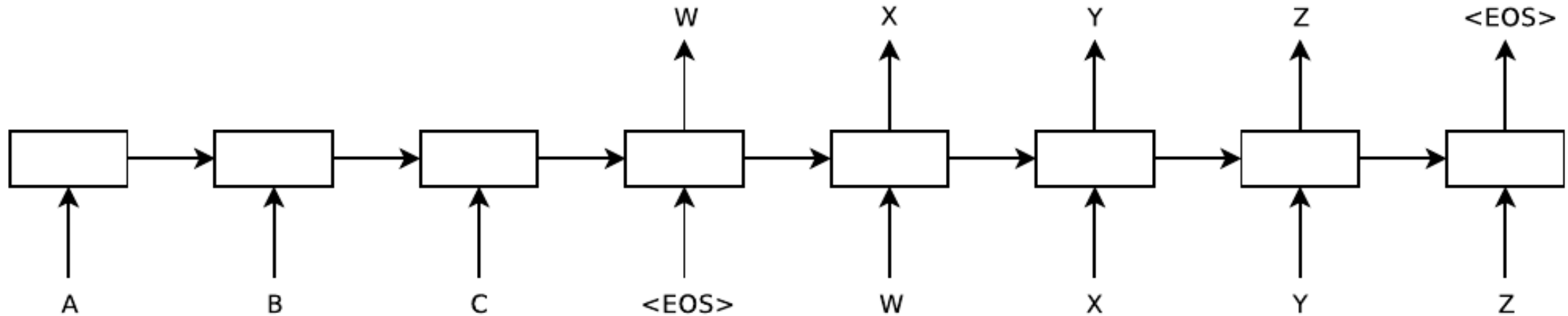
[BOS]: start token

$$p(\text{voiture} \mid \text{La}, x)$$

$$p(\text{rouge} \mid \text{La voiture}, x)$$

Encoder-decoder models

Use an RNN to summarize the source.
Condition an RNN language model on that summary.

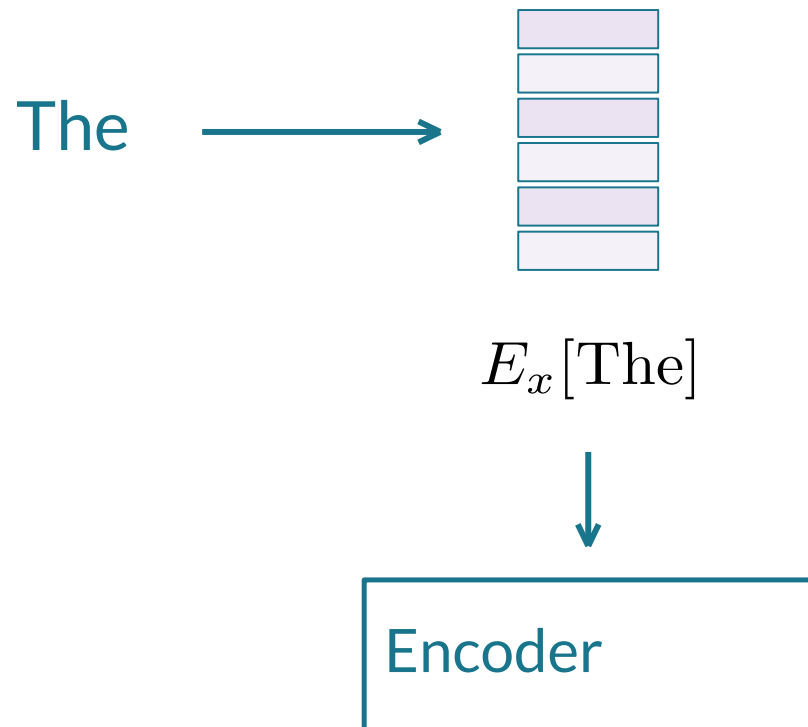


Encoder: source summary

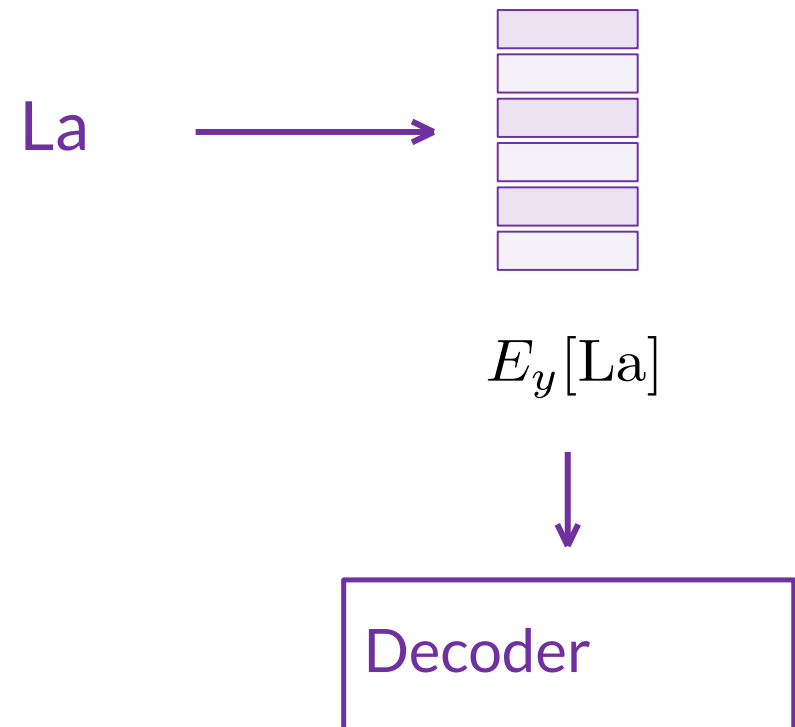
Decoder: target language model

Source and target embeddings

Source vocabulary

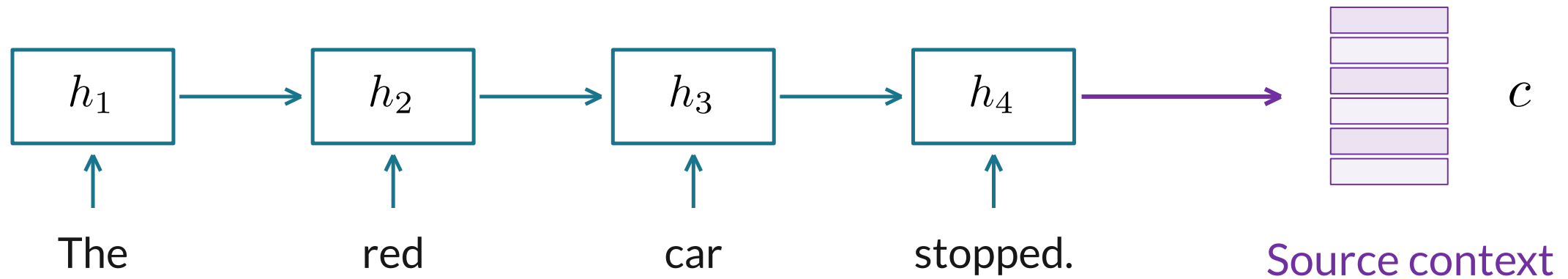


Target vocabulary



E_x, E_y : learned tables. Brackets select a token's vector.

The encoder



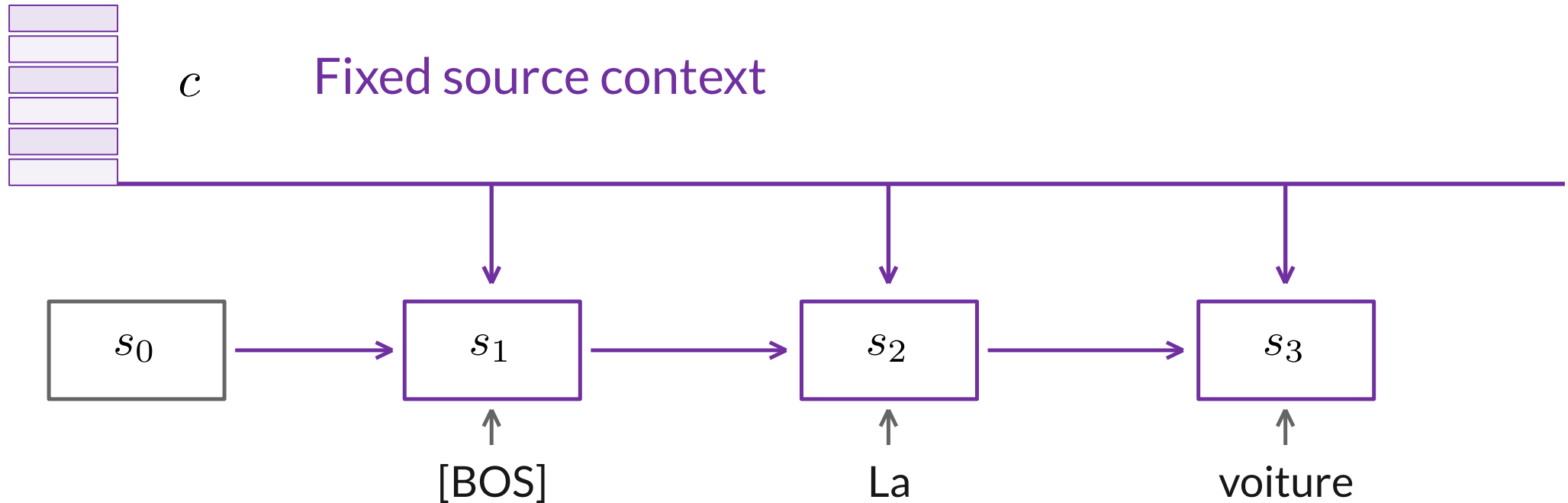
$$h_j = f_{\text{enc}}(h_{j-1}, E_x[x_j])$$

Read one source position

$$c = h_S$$

Keep the final encoder state

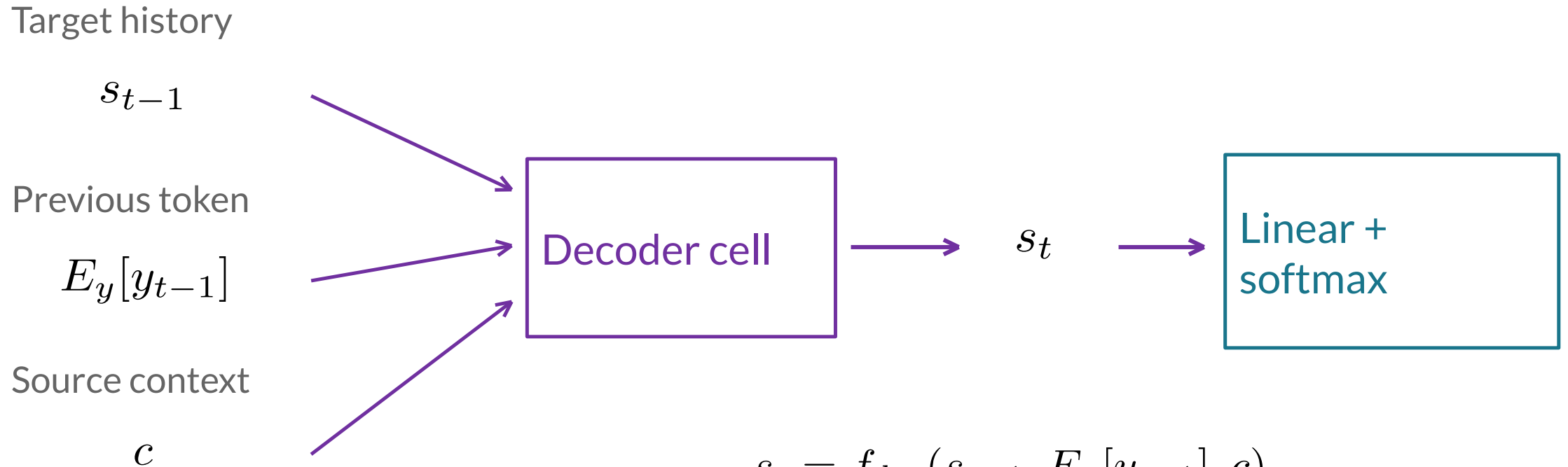
The decoder state



$$s_0 = g(c)$$

g : maps source context
to the initial decoder state

One decoder step



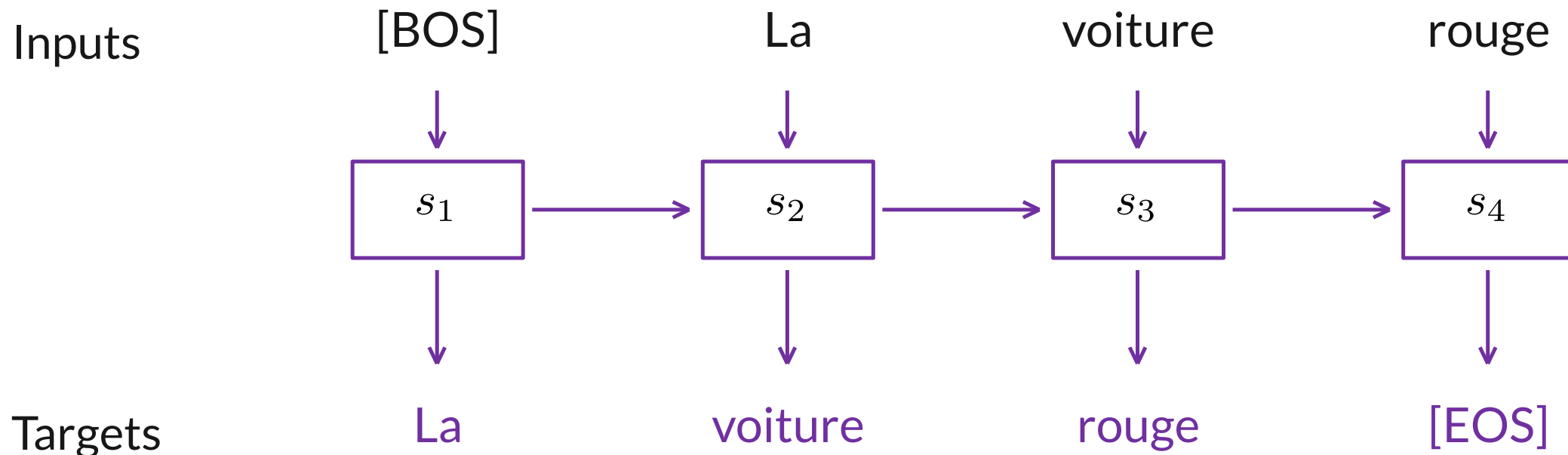
$$s_t = f_{\text{dec}}(s_{t-1}, E_y[y_{t-1}], c)$$

$$p_t = \text{softmax}(W_o s_t + b_o)$$

Probabilities over the target vocabulary

Training inputs and targets

Source: The red car



Teacher forcing: supply the observed previous target token.

Conditional cross-entropy loss



One target token

$$\ell_t = -\log p_{\theta}(y_t \mid y_{<t}, x_{1:S})$$

Sequence loss

$$\mathcal{L} = \sum_{t=1}^T \ell_t$$

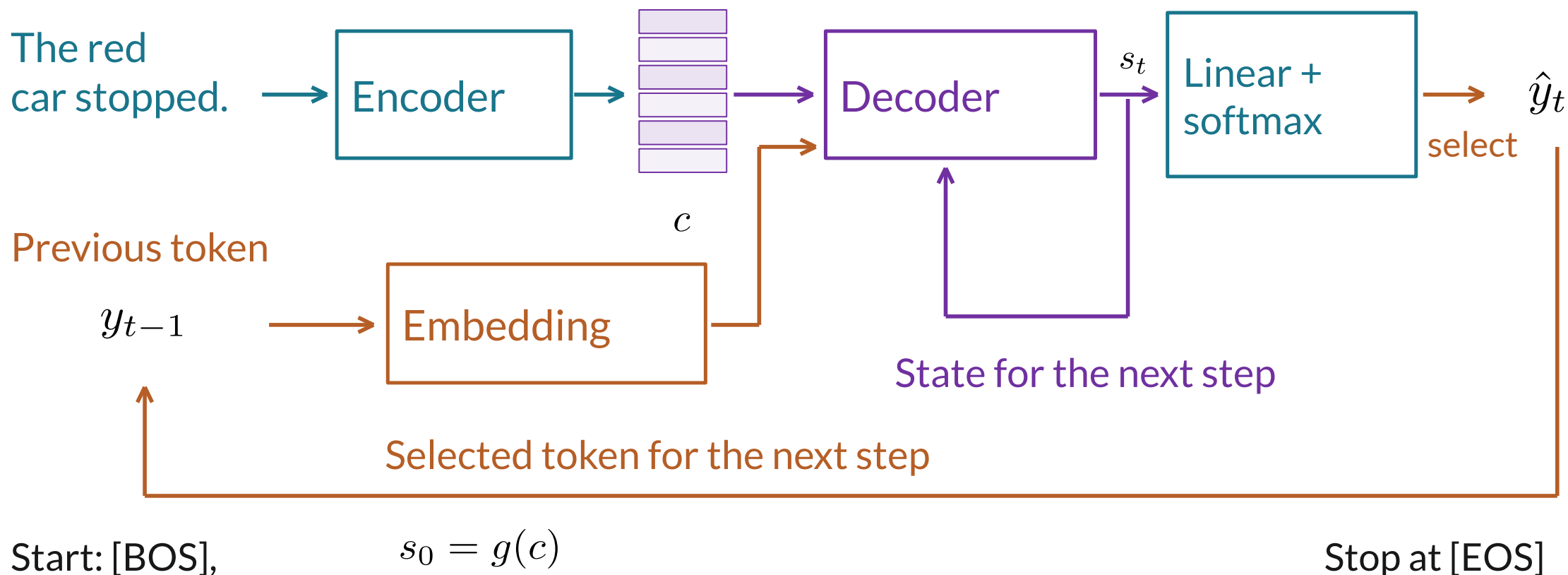
Include [EOS] (end of sequence) in the targets.

For a minibatch, average over valid positions and ignore padding.

Generating the target sequence

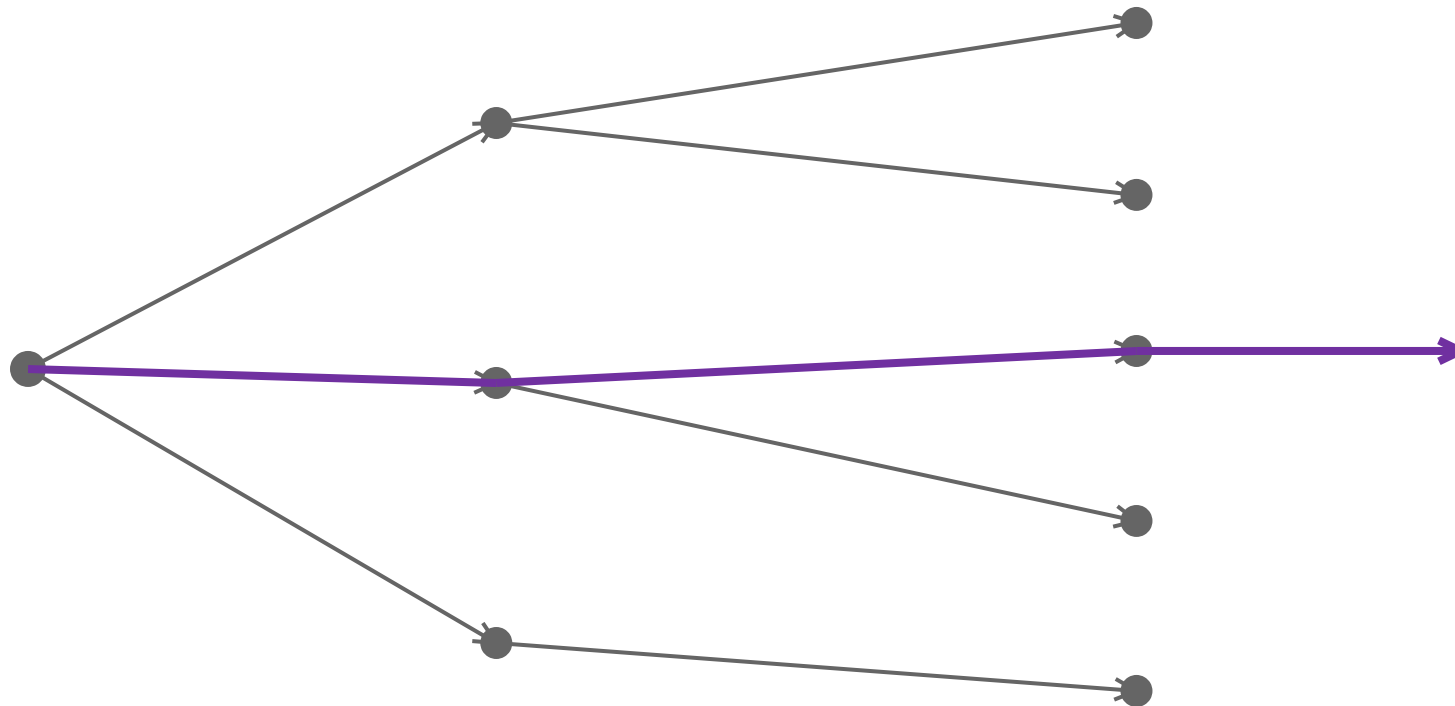
Encode the source once

Generate the target



Decoding a sequence

$$\hat{y} = \arg \max_y \log p_{\theta}(y \mid x)$$



Each branch adds
one token.

A complete
output sequence

argmax: choose the complete sequence with the highest model probability.

Greedy decoding can miss the best sequence

Two-token translation of “Goodbye.”



Greedy: choose the most probable next token at each step.

Beam search

Keep 2 prefixes

À 0.6

Au 0.4

Expand both

À bientôt

À demain

Au revoir

Au printemps

0.33

0.27

0.36

0.04

Keep the best 2

Au revoir

À bientôt

Rank all extensions together.

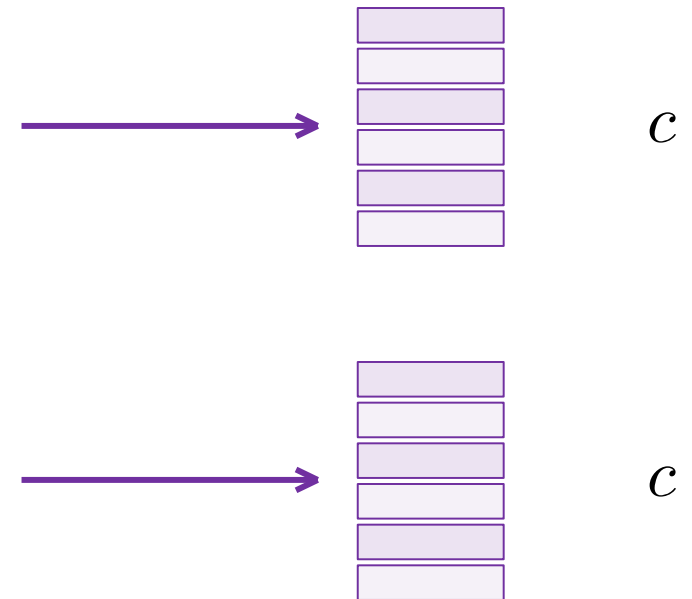
The fixed-summary bottleneck

Source sentence

The red car stopped.

The red car beside the blue van
did not stop.

Fixed-size context



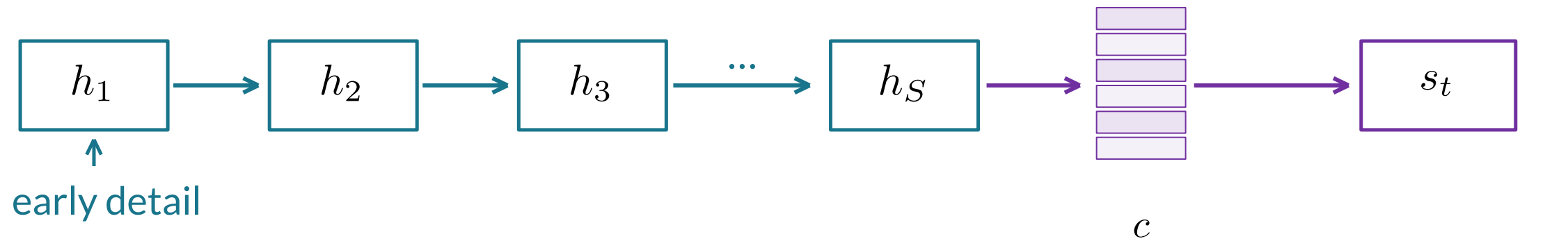
More source details, the same number of stored values.

Every output step must recover the details it needs from this one vector.

Long paths through the encoder

An early source detail must survive many encoder updates.

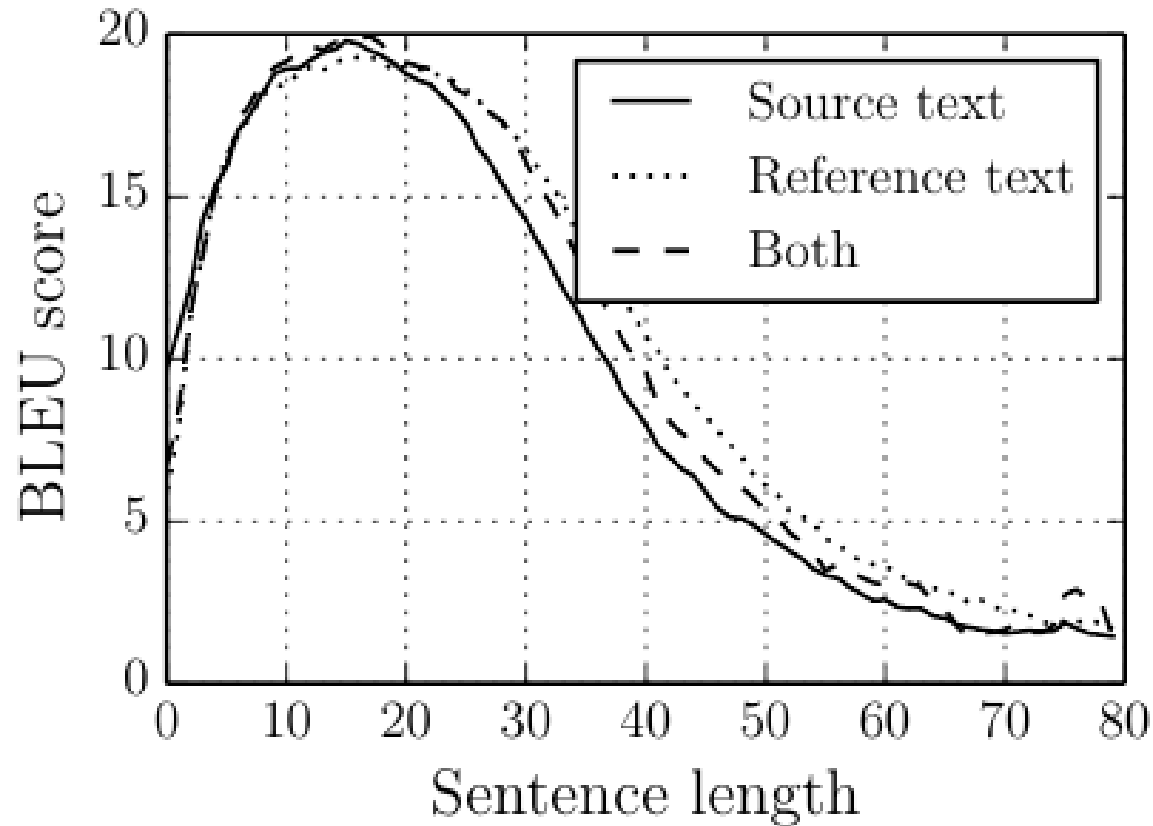
Encoder states



Relevant details can be weakened or overwritten along this path.

Training must also propagate gradients through this long encoder chain.

Longer sentences in early seq2seq models



Same model, grouped by source, target, or both sentence lengths.

An early recurrent encoder-decoder

BLEU: translation score (higher is better)

This system's quality dropped on longer sentences.

Source information at each output step

English source

The

red

car

Predicting the noun

Predicting its color

La voiture

La voiture rouge

voiture = car

rouge = red

The decoder needs different source details at different output steps.

Lecture outline

01 Sequence-to-sequence models

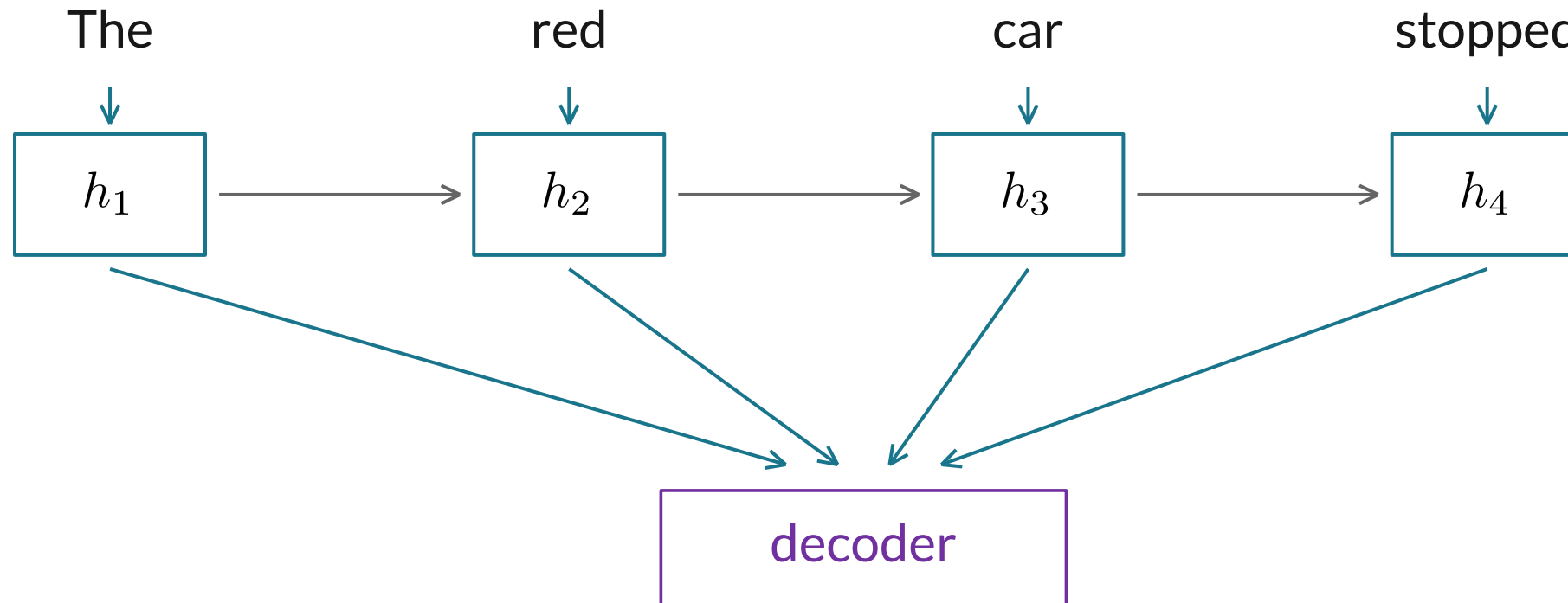
| 02 **Attention**

03 Forecasting and image captioning

04 Self-attention

Access to all source states

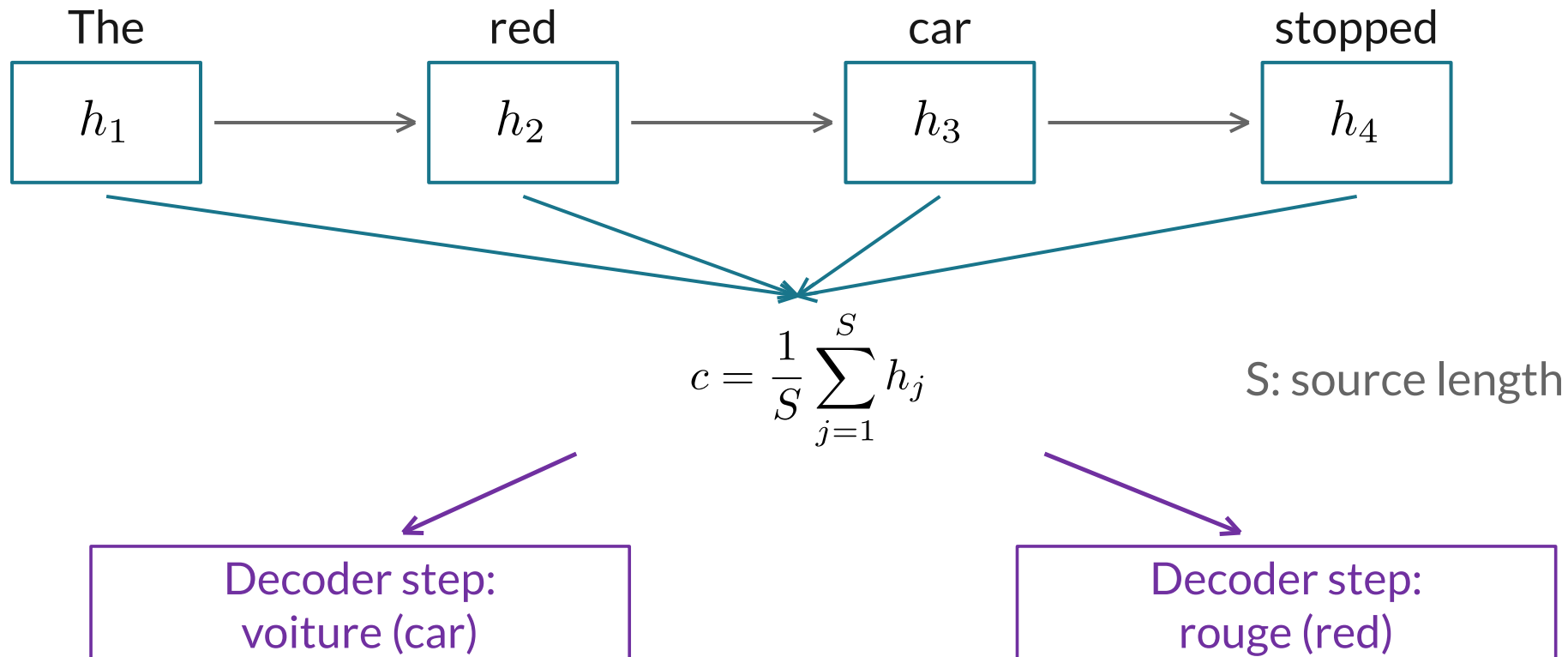
Keep a representation at every source position



Earlier details remain directly available to the decoder.

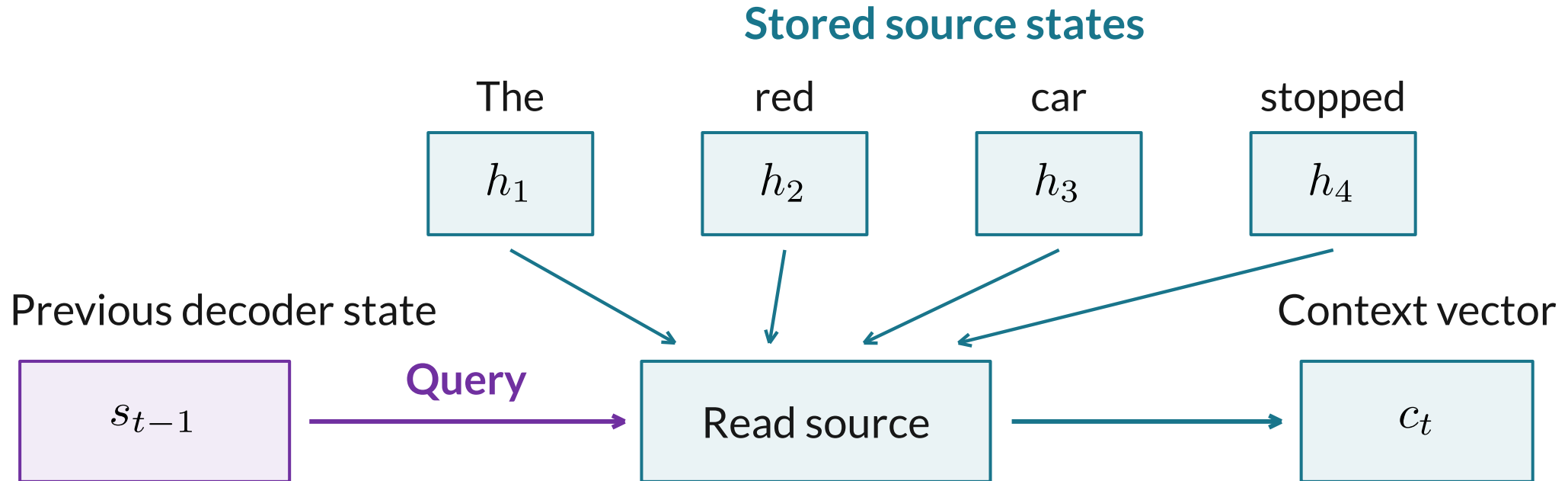
Fixed source averaging

A simple readout: average the stored source states



Different target words need different information.
Let each decoder step choose its own source weights.

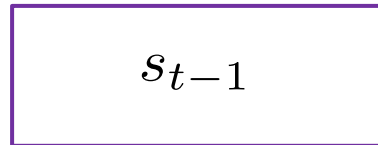
Decoder query



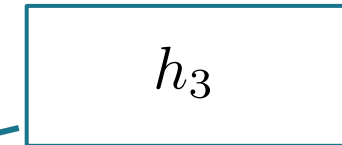
The previous decoder state supplies the query.
It determines which source information is useful for the next prediction.

Learned alignment scores

Previous decoder state



Source state for car



Higher score: stronger match to the decoder state

$$e_{tj} = v_a^T \tanh(W_s s_{t-1} + W_e h_j + b_a)$$

W_s, W_e

Learned projections

b_a

Learned bias

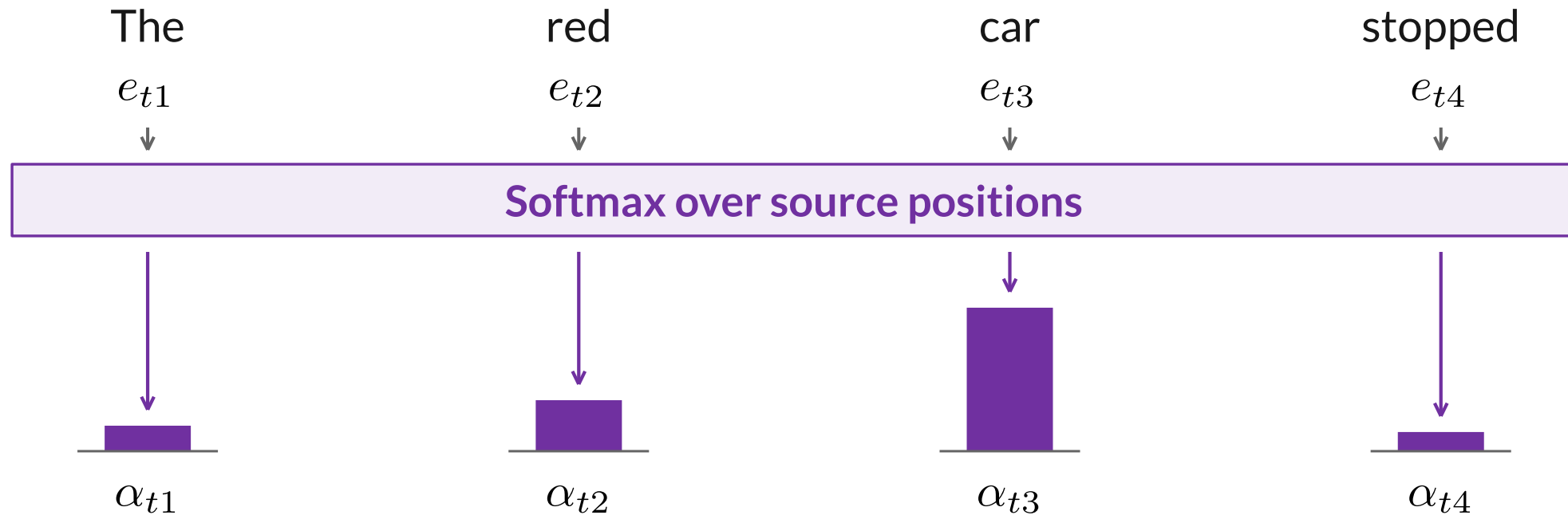
v_a

Learned output weights

t: target step j: source position (car is j = 3)

Attention weights

Example weights when predicting voiture (car)



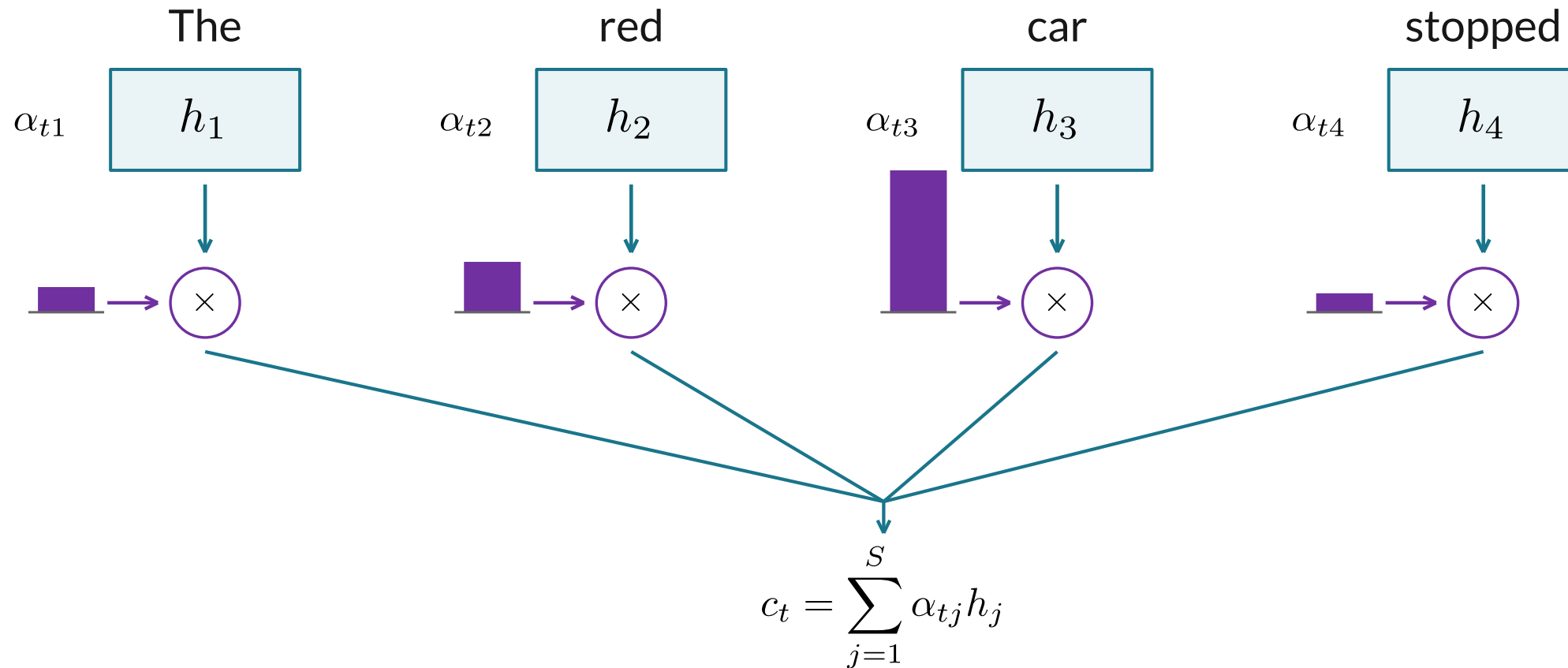
$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{r=1}^S \exp(e_{tr})}$$

$$\sum_{j=1}^S \alpha_{tj} = 1$$

t: target step j: source position S: source length

The context vector

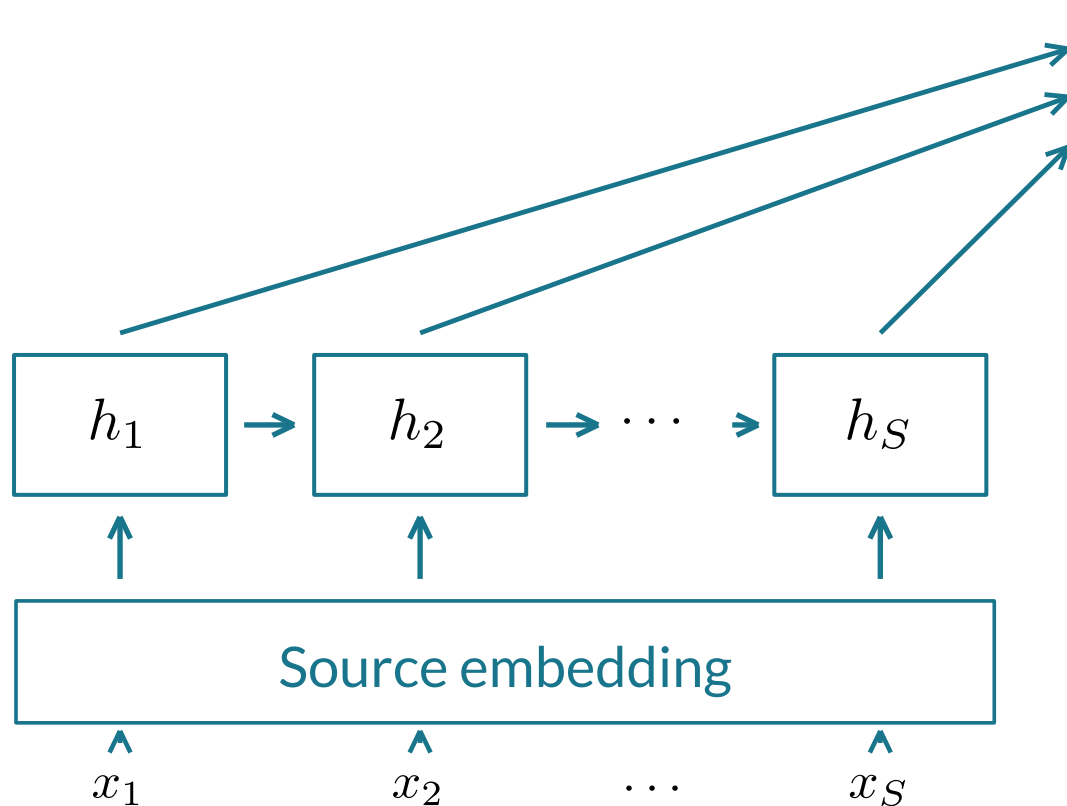
Example weights applied to the source vectors



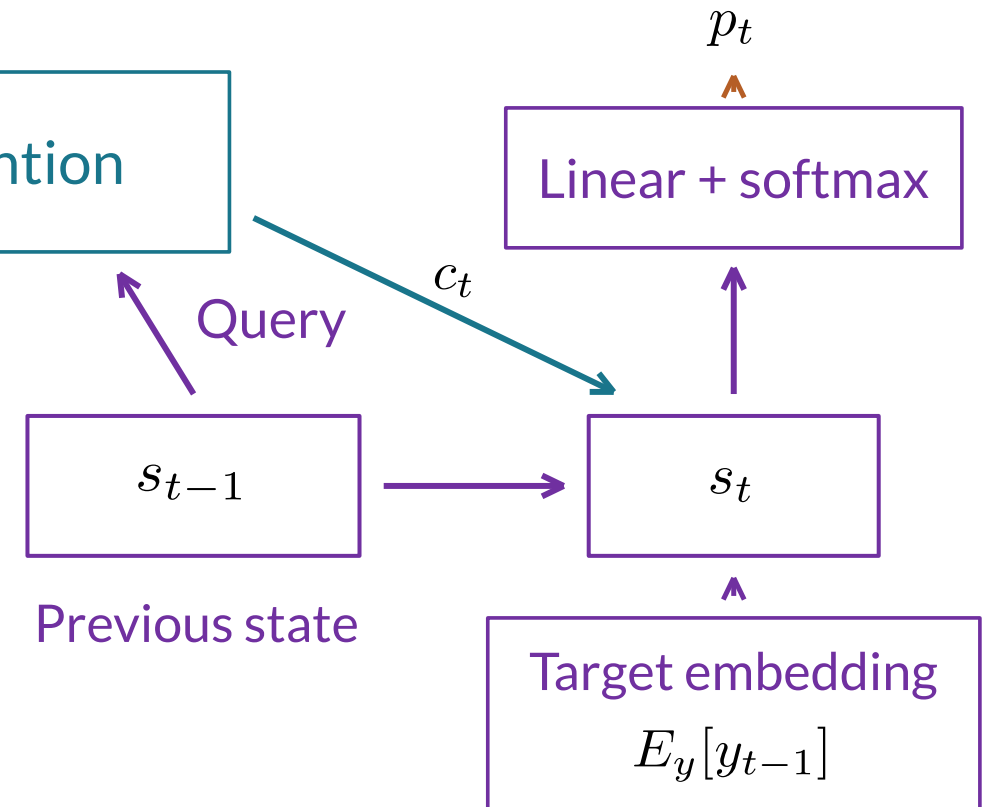
Each weight scales the whole source vector.

Attention in the decoder

Encoder



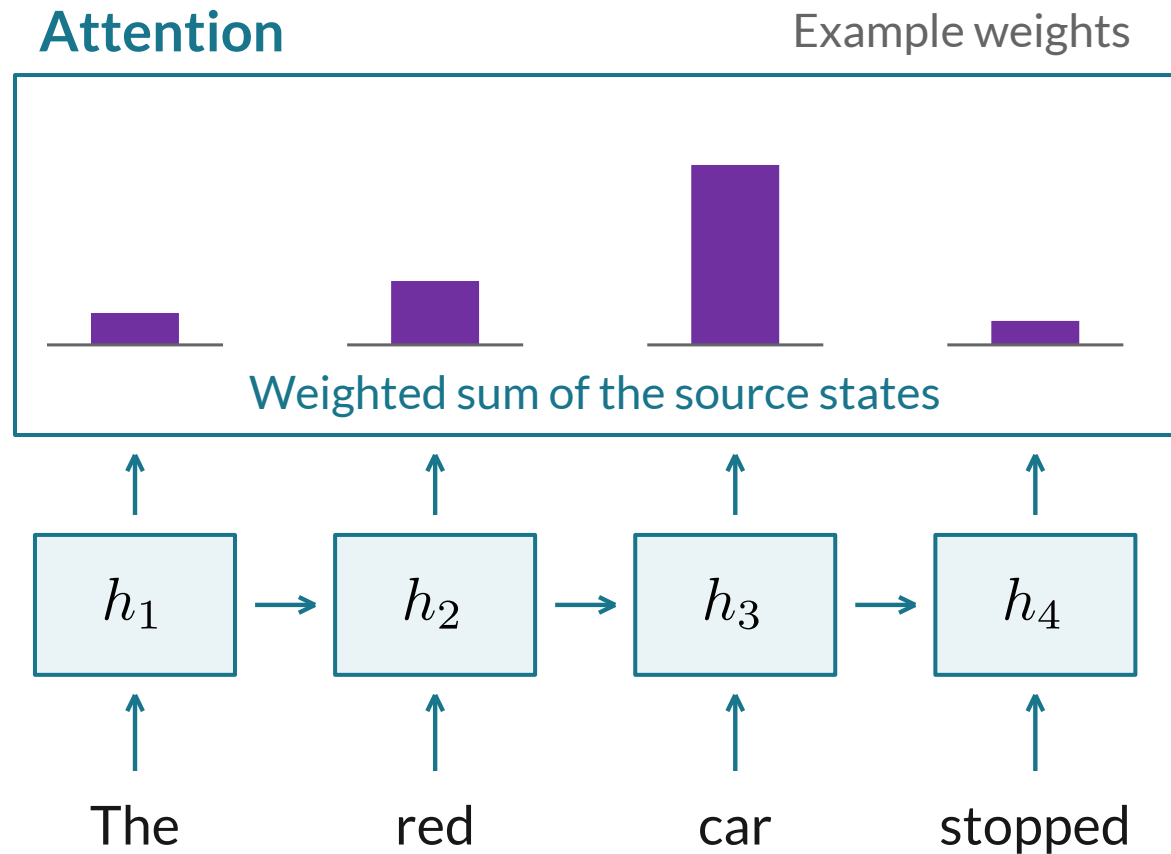
Decoder



Next step: carry s_t forward.

Attention during decoding

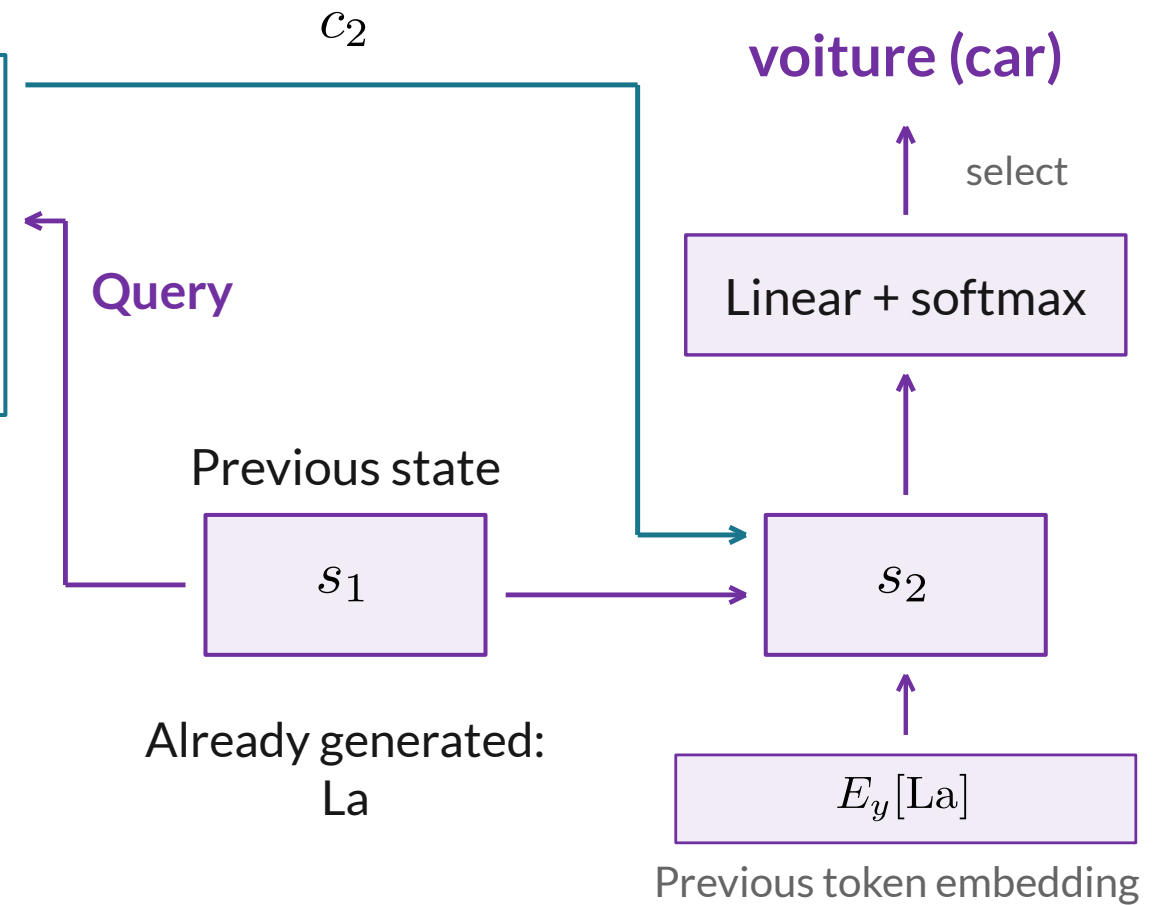
English source



The stored source states stay fixed.

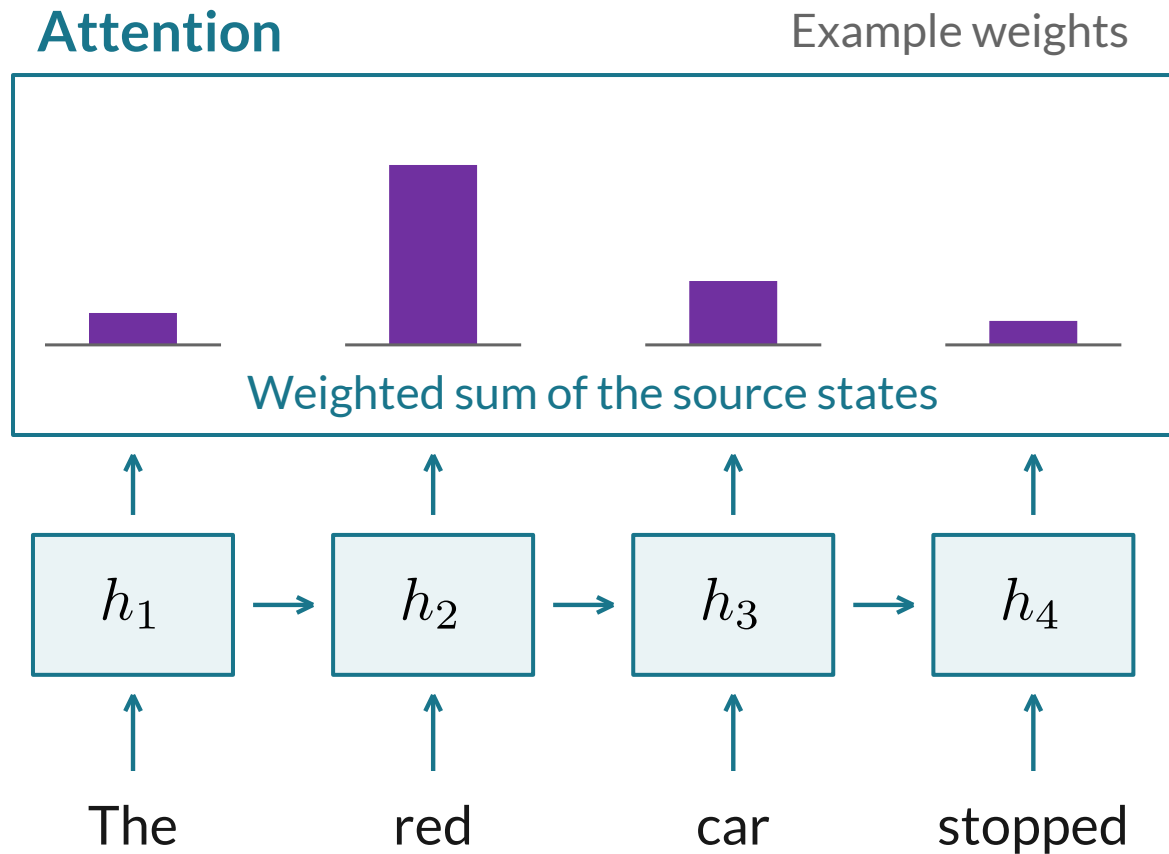
Adapted from Stanford CS231n

French target • step 2



Attention across decoding steps

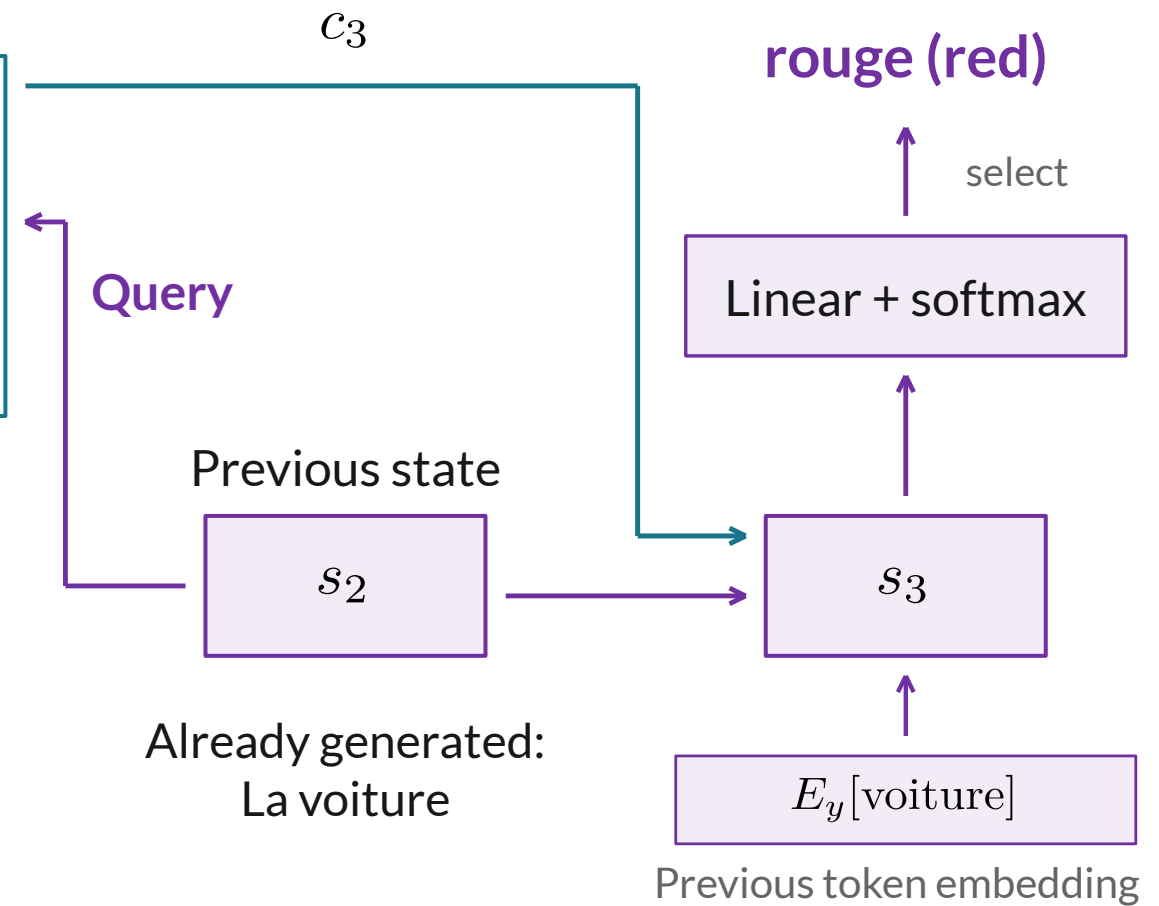
English source



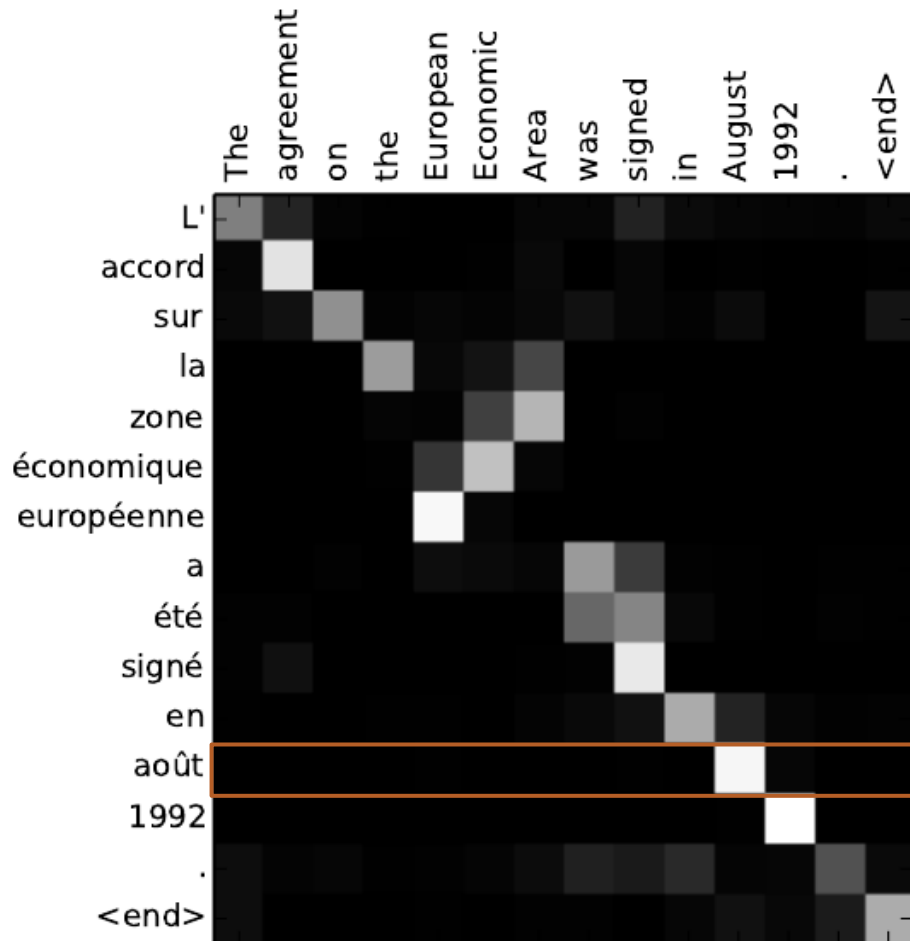
The stored source states stay fixed.

Adapted from Stanford CS231n

French target • step 3



Learned alignments



Columns: English source

Rows: French output

Predicting août (August) uses this row of weights.

■ smaller weight
□ larger weight

Reordered source information

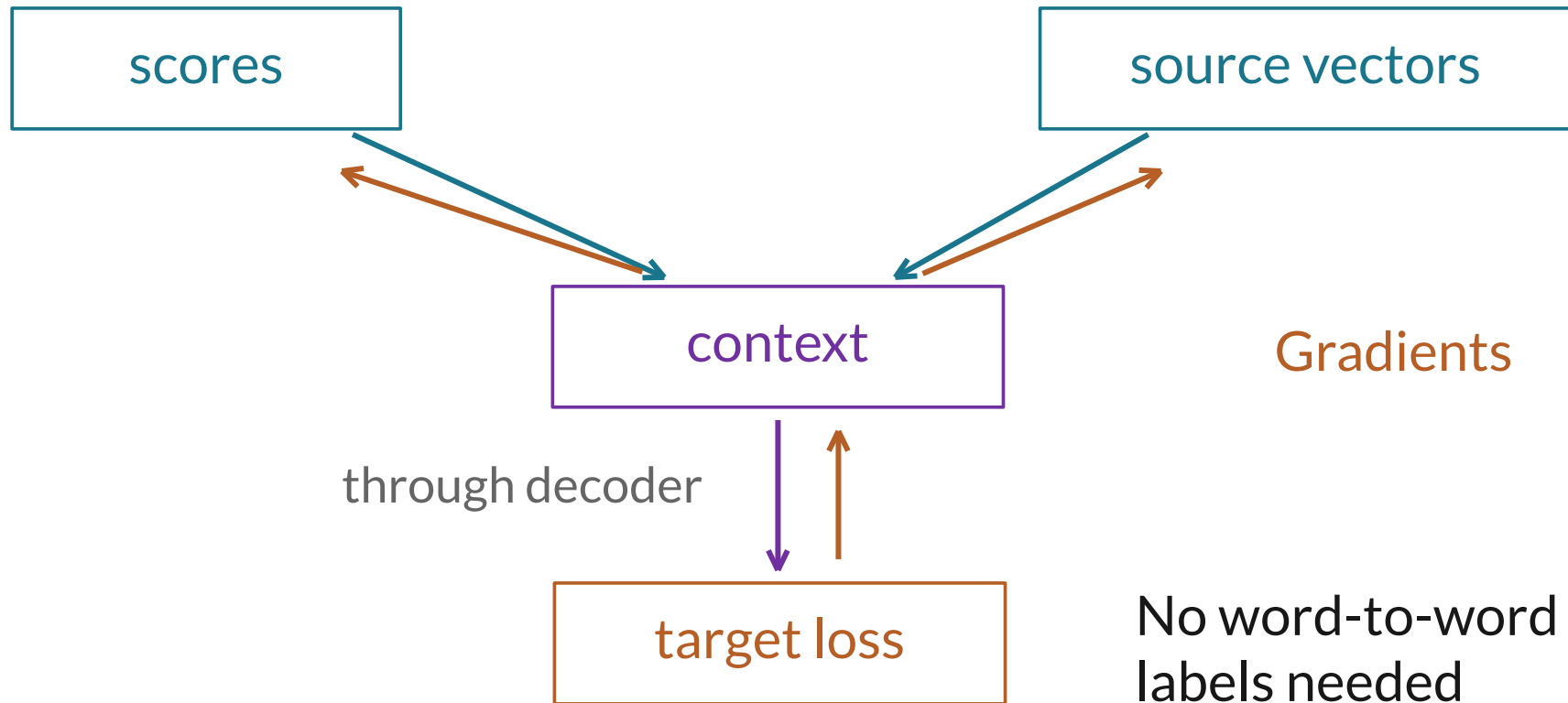
French output	English source		
	European	Economic	Area
zone (area)			
économique (economic)			
européenne (European)			

Output order continues downward.

Source emphasis moves right to left.

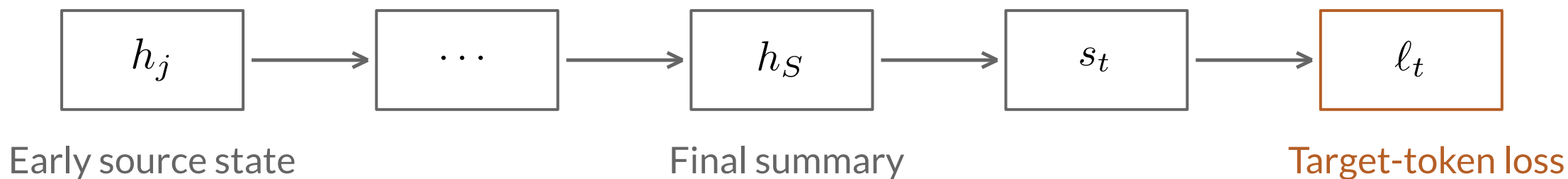
Learning attention

Soft weights give a differentiable readout.
Backpropagation trains the scoring network.



Direct source access

Fixed summary



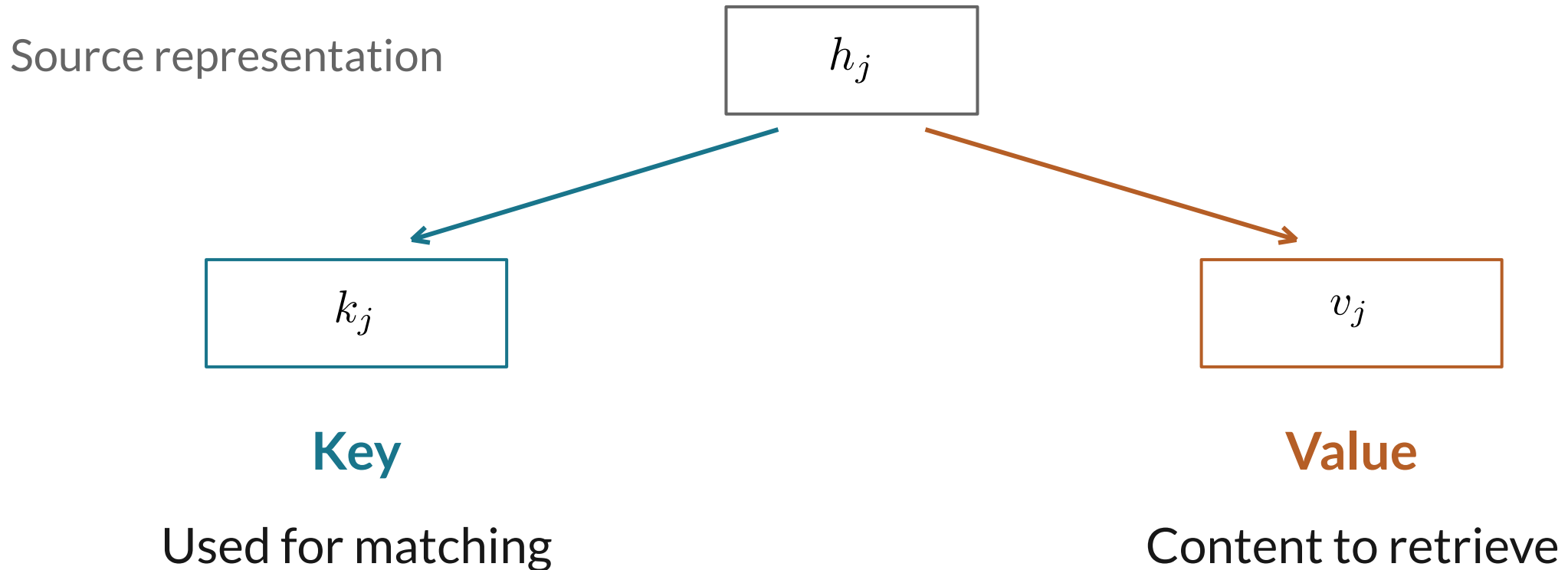
Attention read



Source information and gradients can bypass the final summary.

Keys and values

Features useful for matching can differ from the content to retrieve.



Learned Q/K/V projections

Decoder state

$$q_t = W_Q^T s_{t-1}$$

Query

Match with keys

Source state

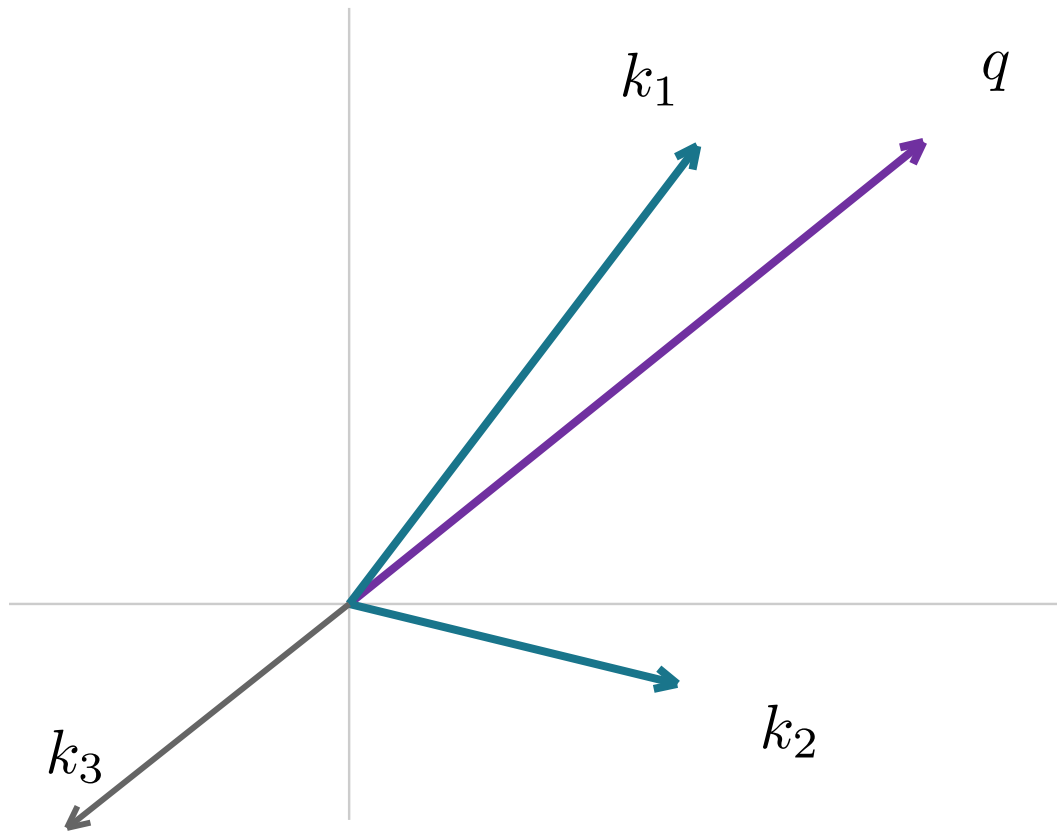
$$k_j = W_K^T h_j$$

$$v_j = W_V^T h_j$$

W matrices are learned and shared across positions.

Dot-product matching

An alternative to the scoring network: dot products of learned query/key features.



$$e_j = q^T k_j$$

$$= \|q\| \|k_j\| \cos \theta_j$$

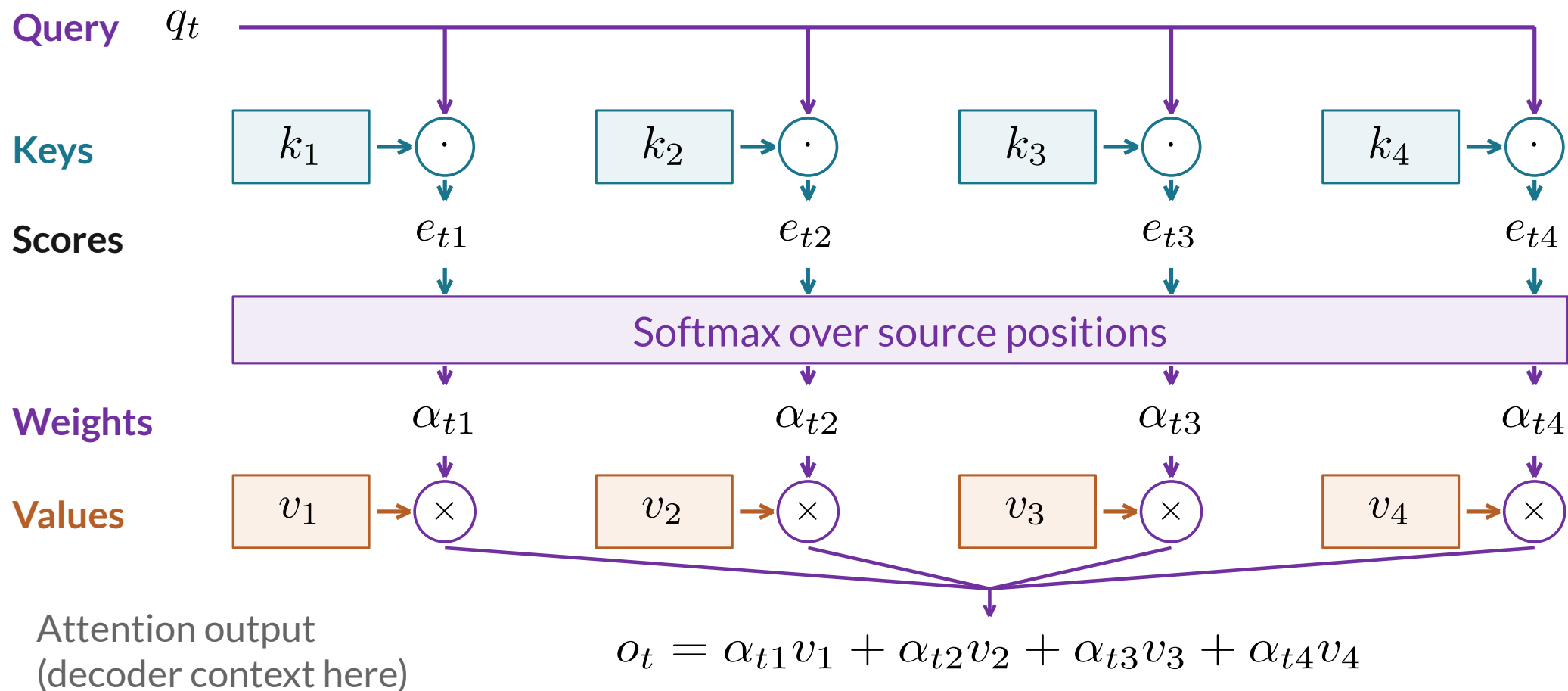
θ : angle between the vectors

The two norms give the vector lengths.

Direction and length affect the score.

One-query attention

Compare the same query with every key.



Parameters and attention vectors

Learned parameters

$$W_Q, W_K, W_V$$

Shared across examples

Computed for an input

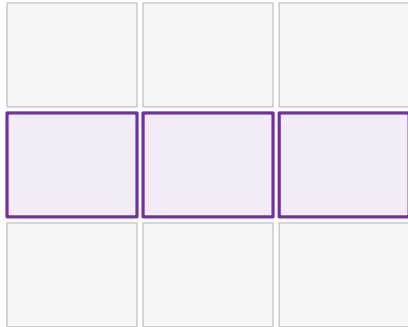
$$q_t, k_j, v_j$$

Change with the source
and decoder state

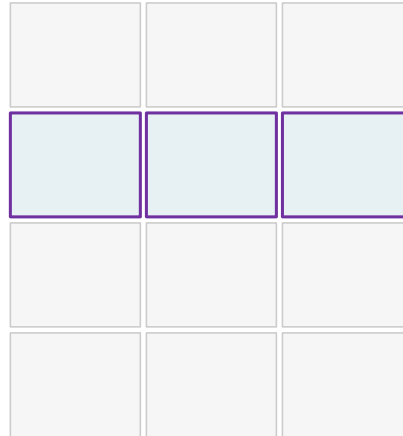
For one source: reuse the keys and values while the query changes.

Multiple queries

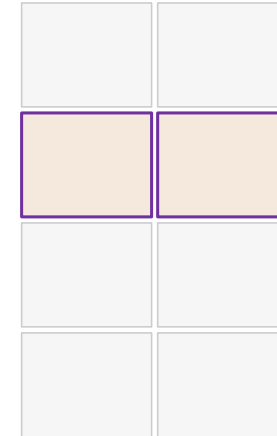
Write each vector as one row.

 Q d_k q_i^T 

query rows

 K d_k k_j^T 

source rows

 V d_v v_j^T 

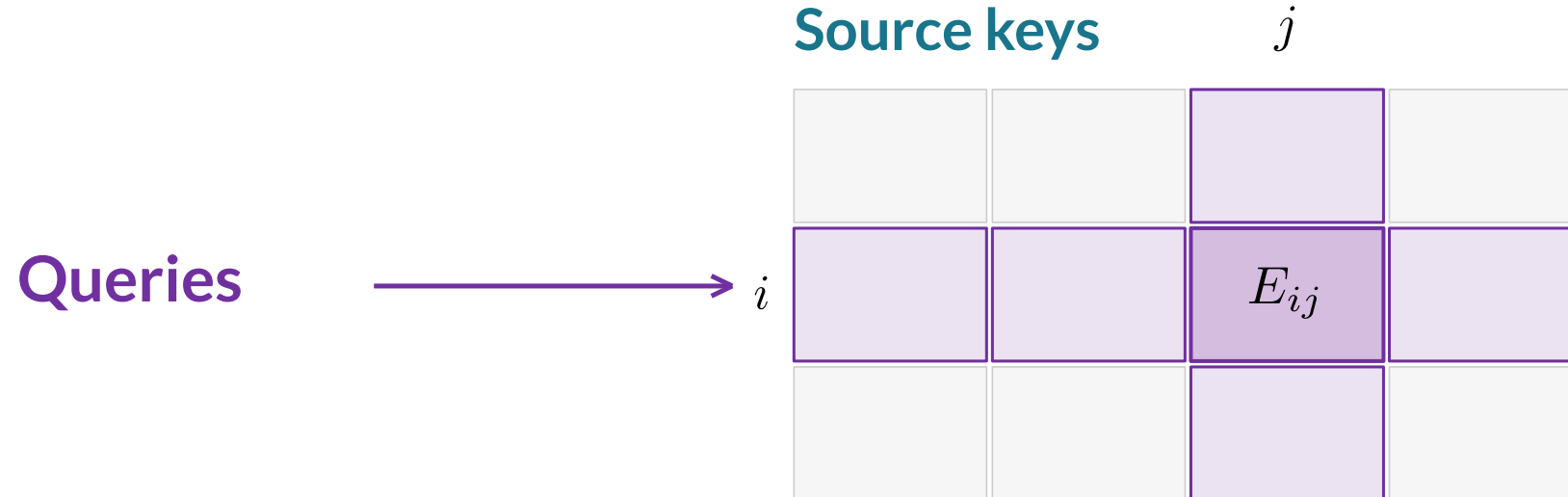
same source rows

d_k : query/key vector length

d_v : value vector length

The score matrix

$$E = QK^{\top}$$

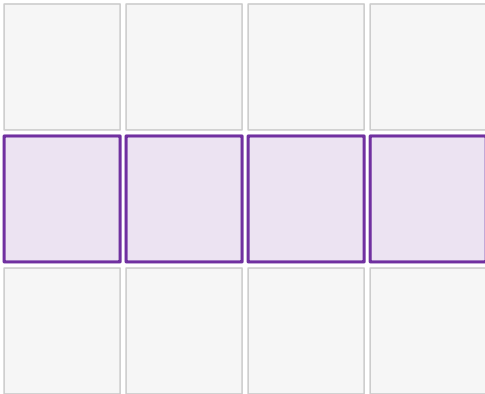


$$E_{ij} = q_i^{\top} k_j$$

Matrix readout

$$A = \text{softmax}_{\text{row-wise}}(E)$$

A



Attention weights
Each row sums to 1

$\times$

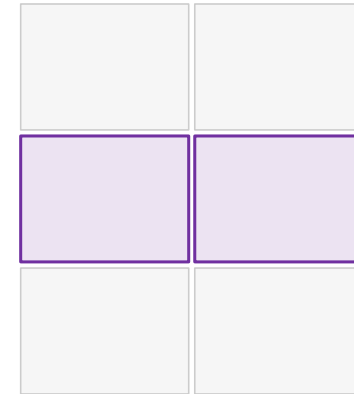
V



Value vectors

$=$

O



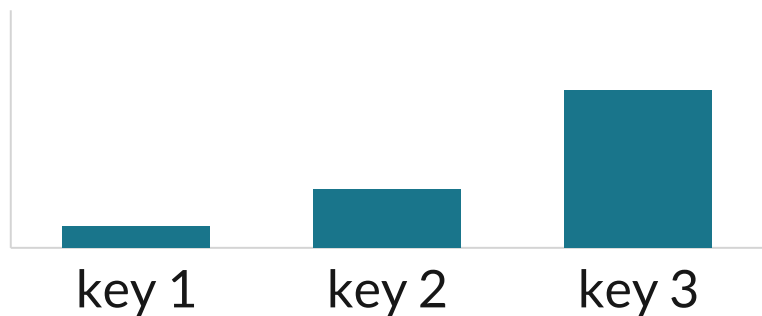
One output
per query

Dot-product scale

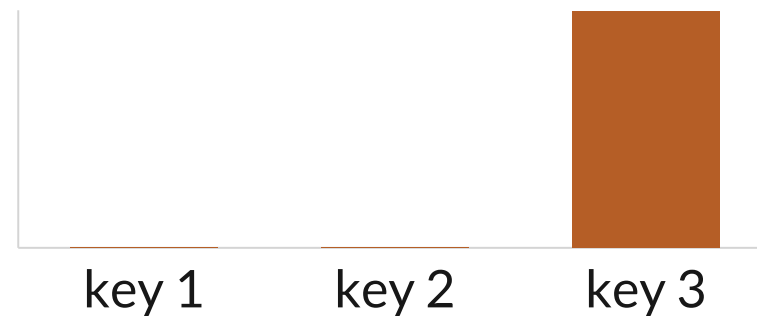
$$q^T k = \sum_{r=1}^{d_k} q_r k_r$$

For roughly independent coordinates of similar scale, score spread grows with the square root of key width.

Attention weights after softmax



Moderate score differences



Larger score differences

With weights near 0 or 1, small score changes barely change the readout.
Learning attention can slow.

Scaled dot-product attention

$$O = \text{softmax}_{\text{row-wise}} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

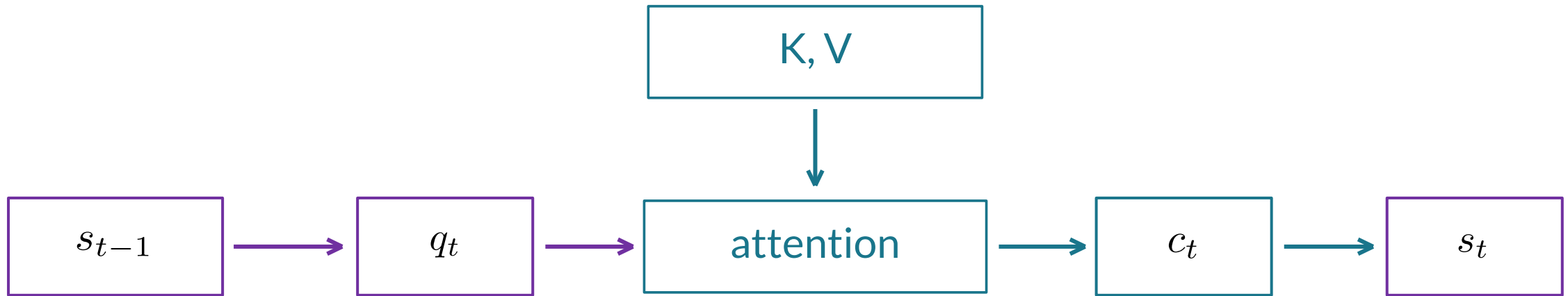
The division counteracts the growth in score spread as key width increases.

```
import math
d_k = Q.shape[-1]           # query/key width
scores = Q @ K.T / math.sqrt(d_k) # scaled matching
A = scores.softmax(dim=-1)   # source dimension
O = A @ V                   # weighted values
```

Q, K, V: one vector per row

Attention and recurrence

Reuse source keys and values



$$q_{t+1} = W_Q^T s_t$$

The next query needs the new decoder state. Decoding is still sequential.

Lecture outline

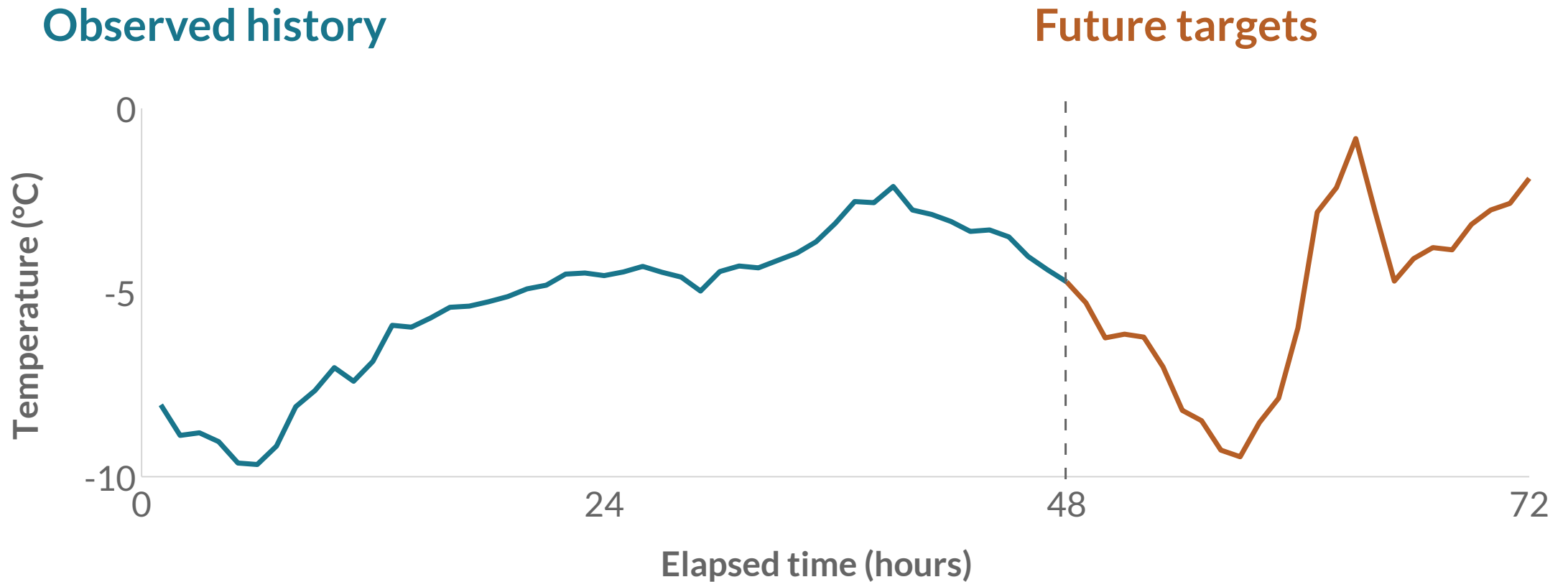
01 Sequence-to-sequence models

02 Attention

| 03 **Forecasting and image captioning**

04 Self-attention

Forecasting future measurements



Use the first 48 hours to predict the next 24 hours.

Inputs and forecasting targets

Observed at each time

$$x_j = \begin{bmatrix} \text{temperature} \\ \text{pressure} \\ \text{humidity} \end{bmatrix}$$

$$x_1, \dots, x_S$$

S observed time steps

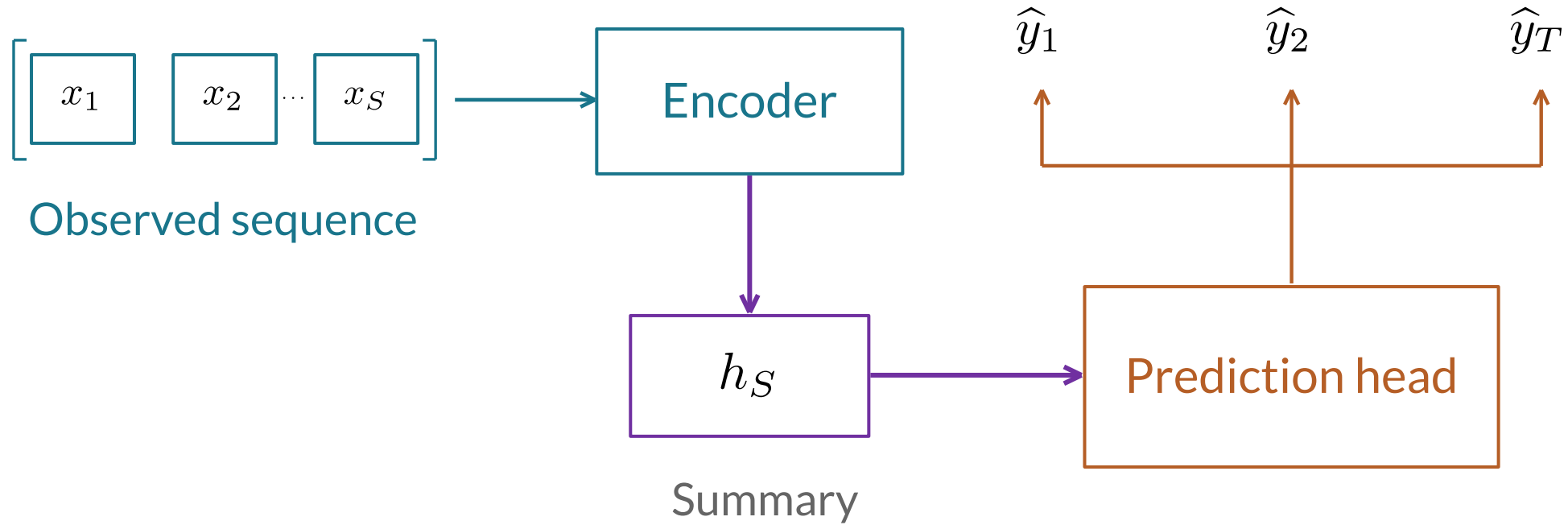
Future target at each step

$$y_t = \text{temperature}$$

$$y_1, \dots, y_T$$

T future time steps

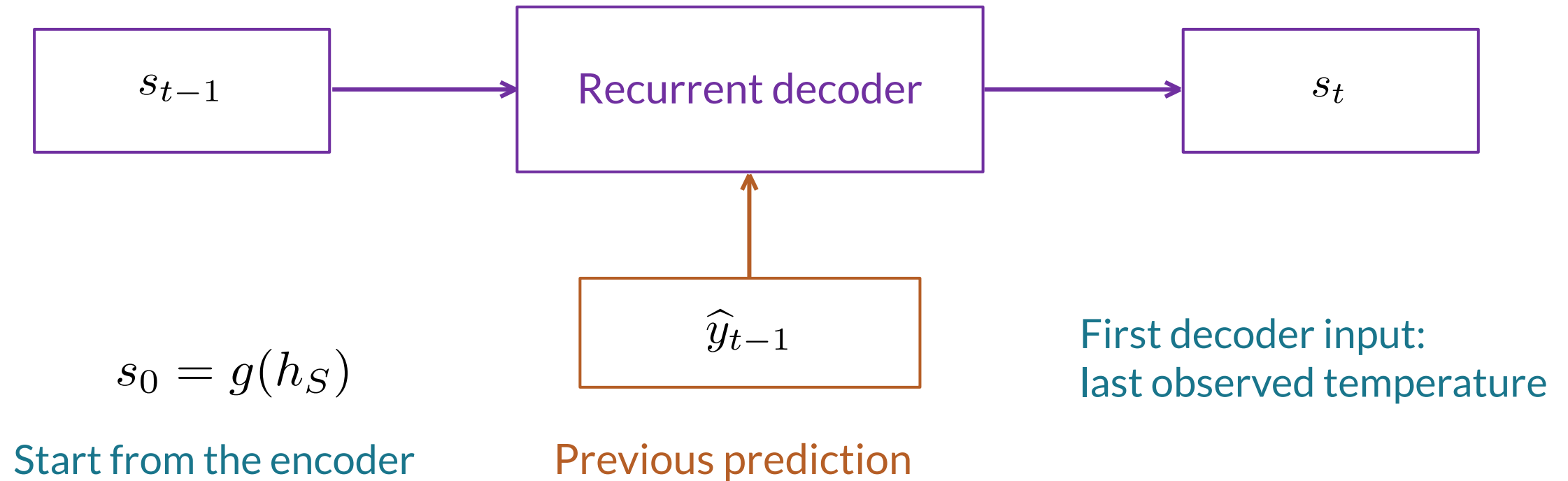
Predicting the whole horizon



One output vector for the full forecast horizon

Recurrent forecasting

One transition, reused across forecast steps



Continuous predictions and loss

A real-valued output

$$\hat{y}_t = w_y^T s_t + b_y$$

Decoder state $\rightarrow$ temperature

Regression over the horizon

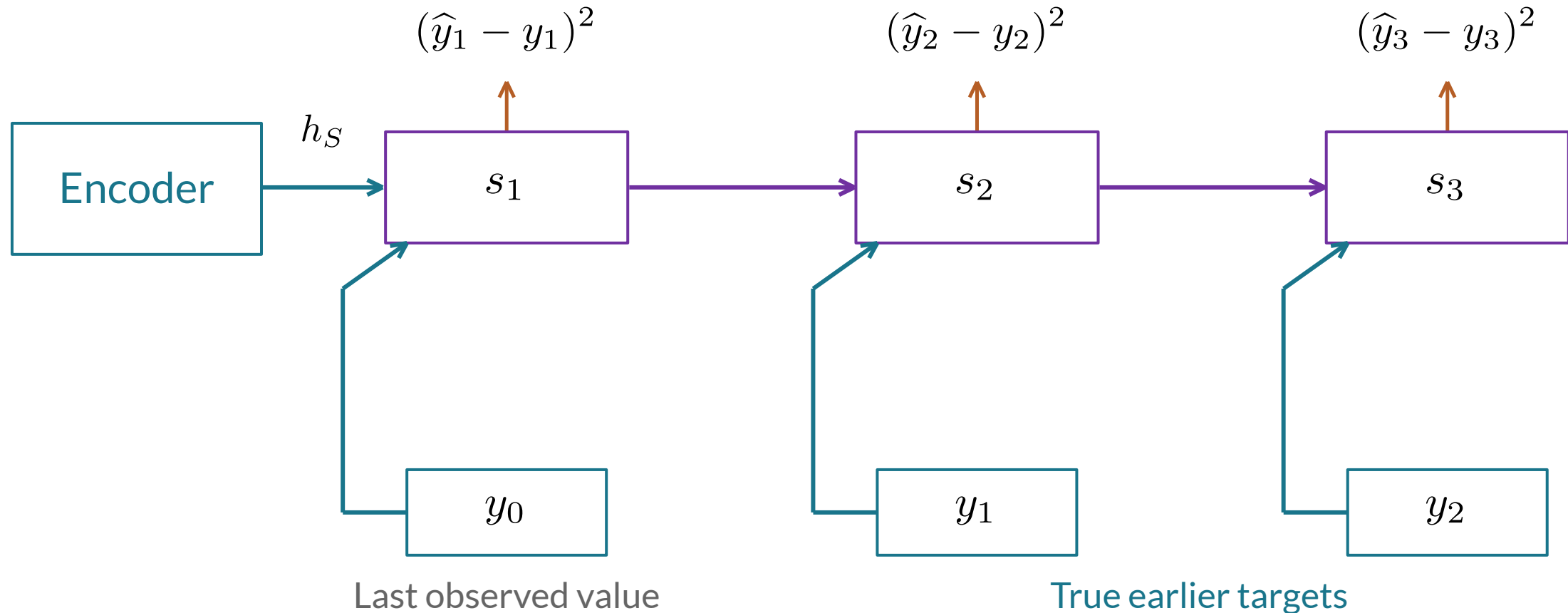
$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^T (\hat{y}_t - y_t)^2$$

$\hat{y}_t$ prediction y_t target

Stop after the chosen horizon T.

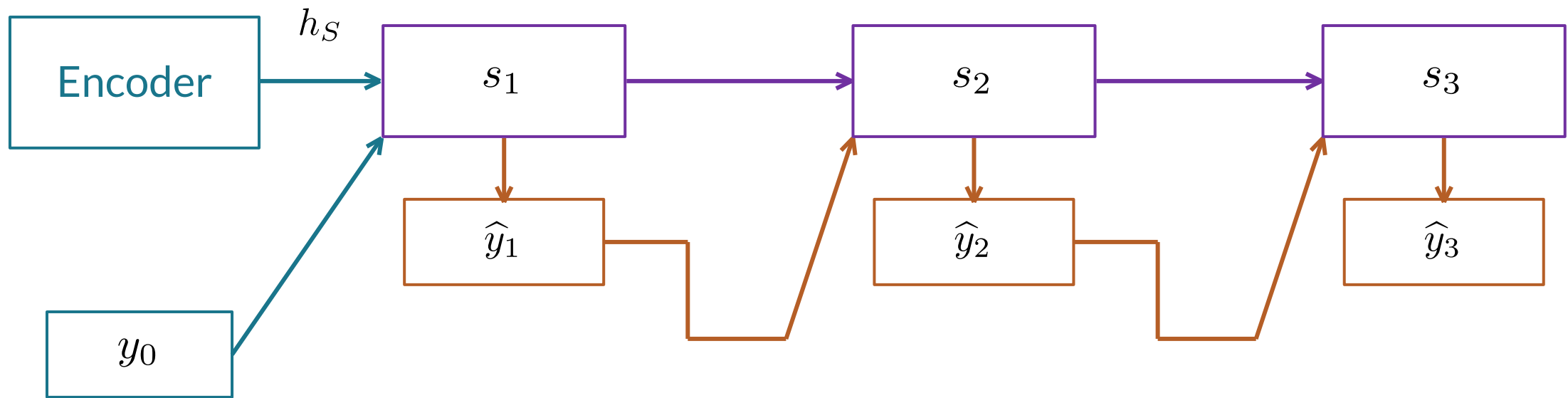
Training with observed targets

Teacher forcing: use the true previous measurement



Recursive forecasting

Predictions become later inputs



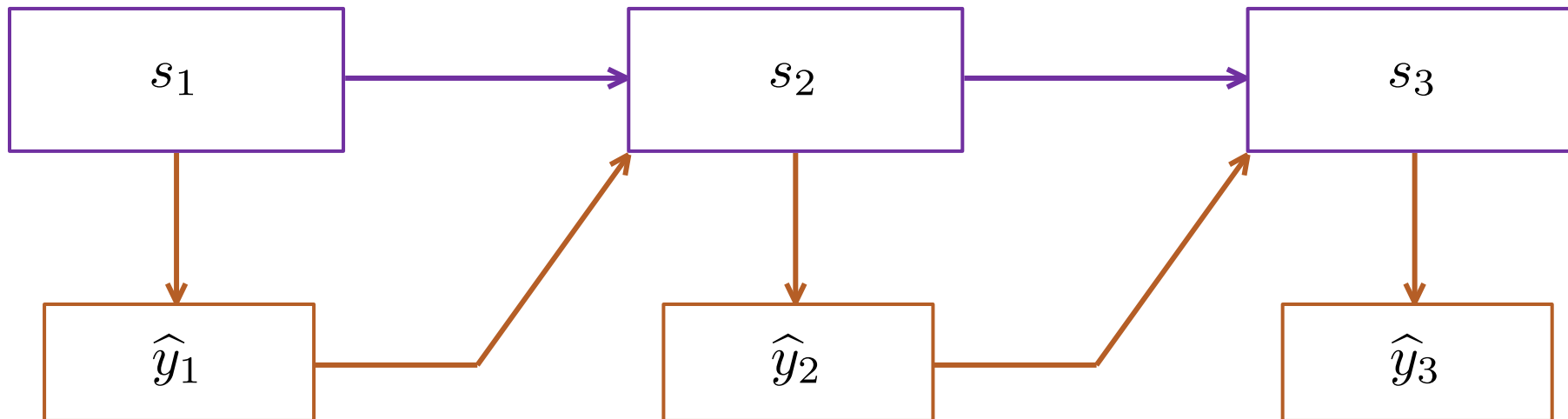
Last observed temperature

Same decoder parameters at every step

Errors fed into later steps

An early error changes later inputs and states

Changed state



Prediction error Changed input

Feedback can amplify, preserve or reduce the disturbance.

Image captioning



Image → sentence

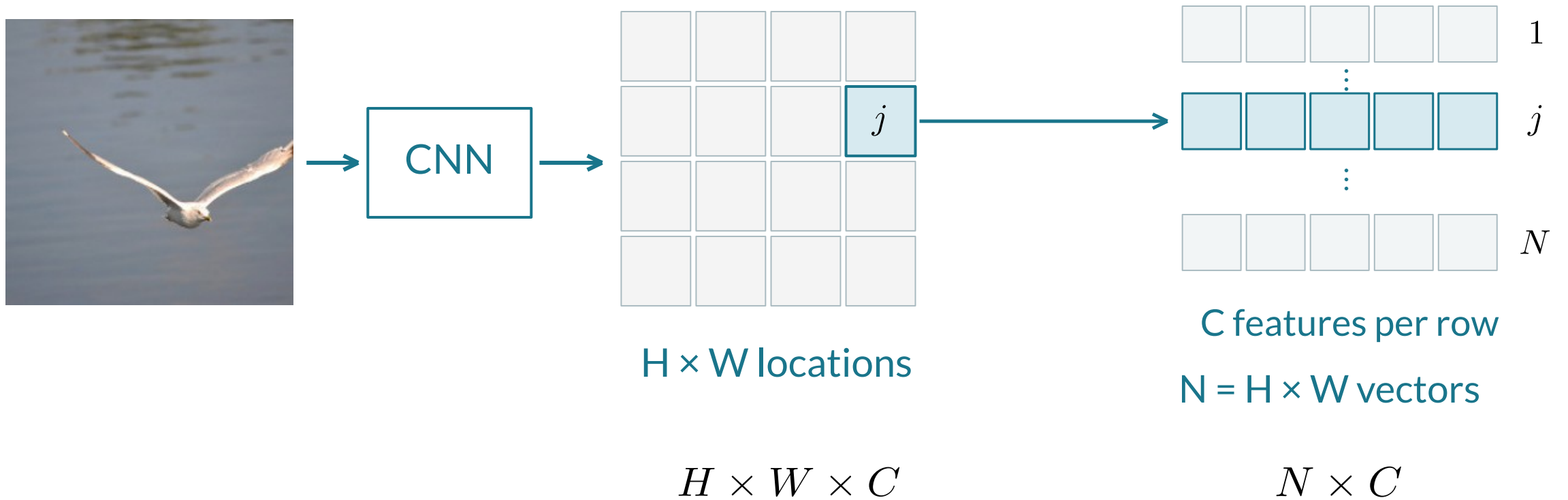


A bird flying over
a body of water.

Objects, actions and relations

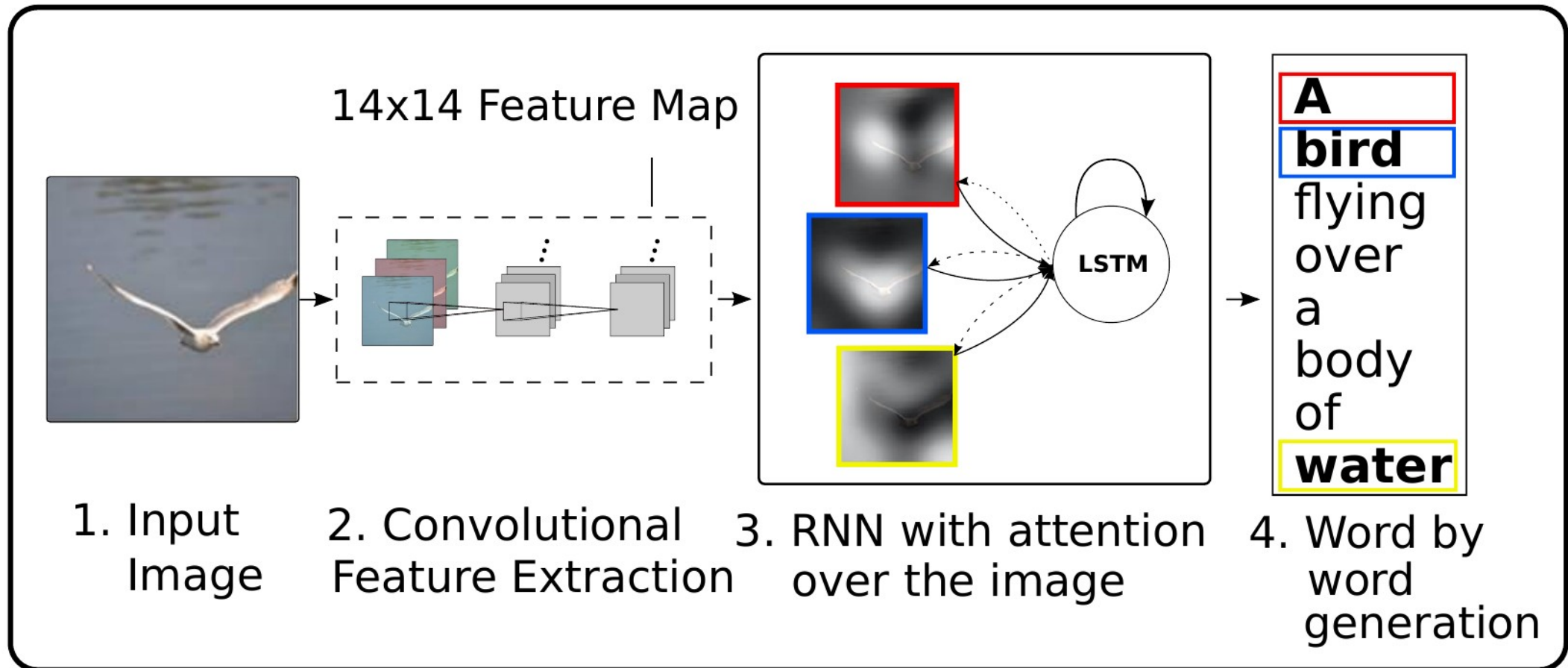
Spatial features as a memory

Each spatial location stores C features



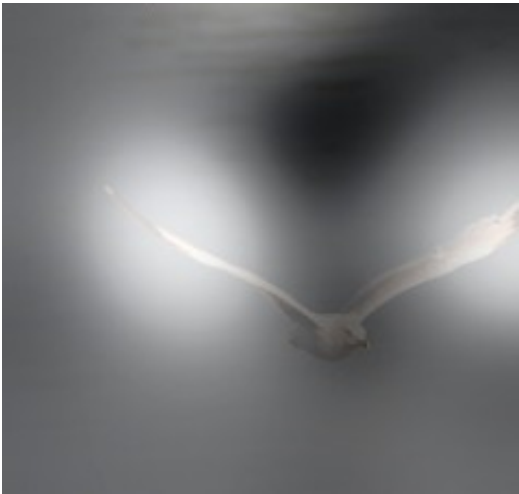
Attention over image regions

The decoder reads different image regions as it generates words.



Word-dependent visual attention

A bird flying over a body of water.



A



bird



flying



water

Brighter regions receive larger attention weights.

Lecture outline

01 Sequence-to-sequence models

02 Attention

03 Forecasting and image captioning

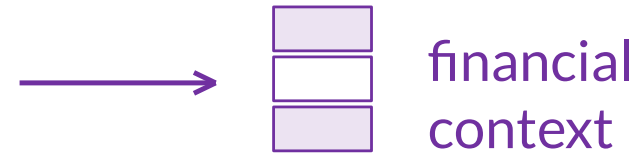
| 04 Self-attention

Context inside a sequence

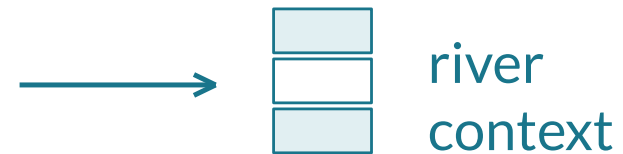
A token's useful features depend on its context.

She deposited money at the bank.

Features for bank



He rested beside the river bank.

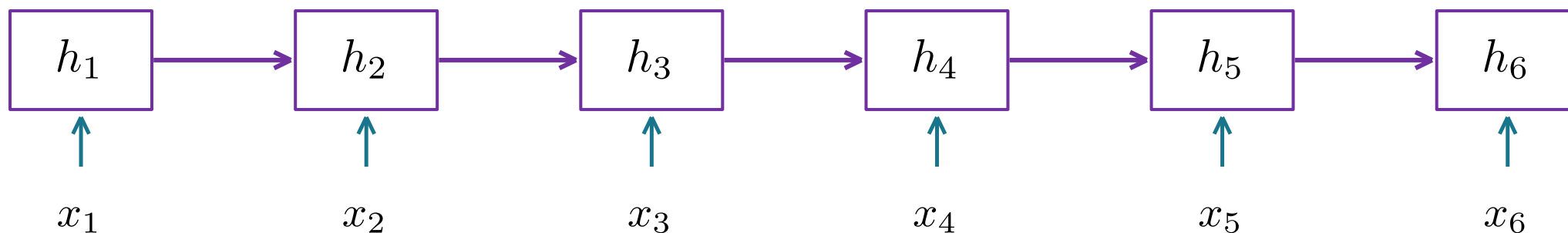


RNNs already build contextual features through hidden states.

Context through recurrent states

Our translation model uses attention to read the source.

Its source features are still built by an RNN.



A fixed-size state must carry context through many updates.

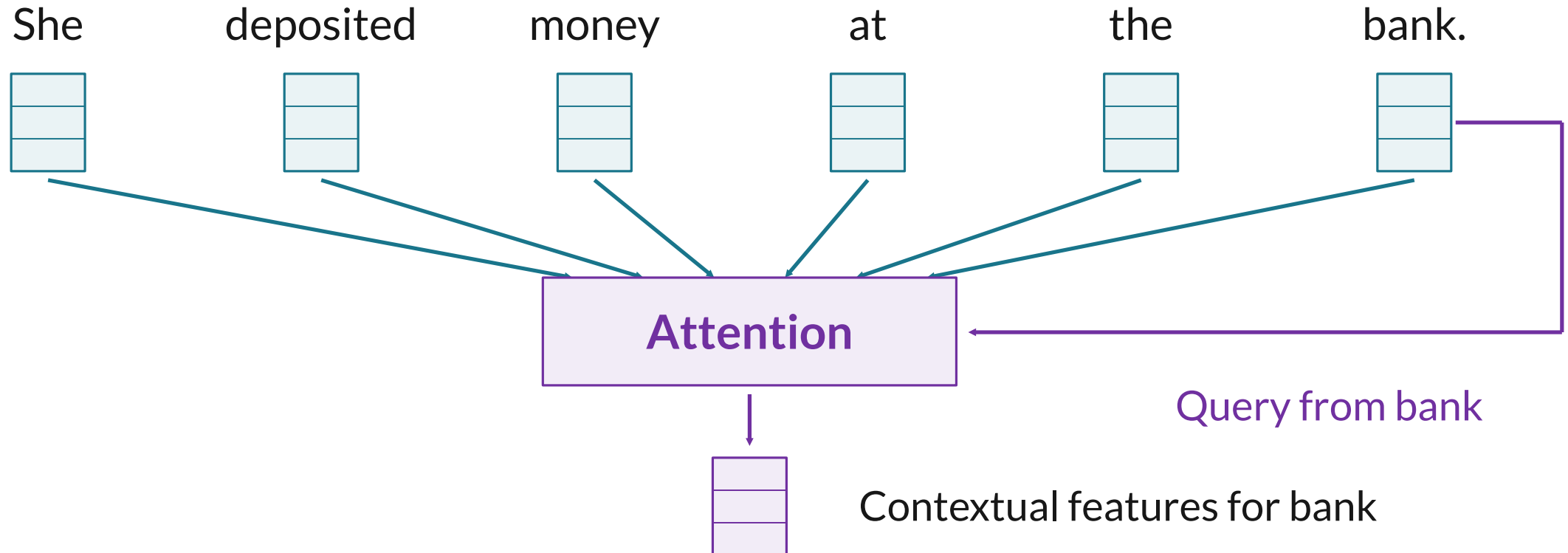
Long paths make learning distant dependencies harder.

Each state also waits for the previous state.

Direct interaction between tokens

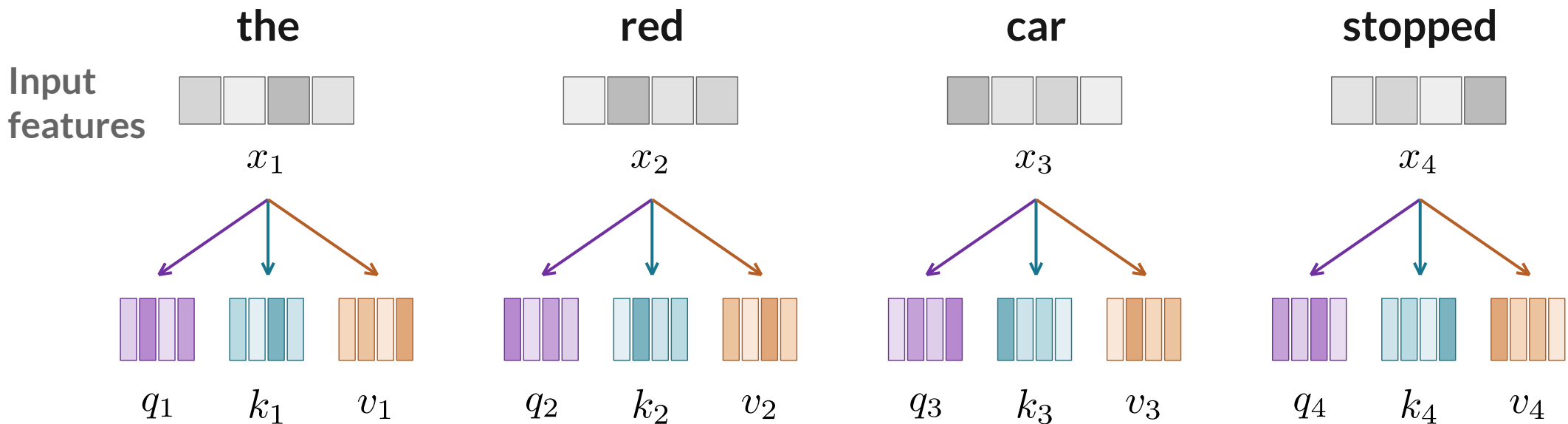
Use each token's features to query the same sentence.

Keys and values from the sentence



Queries, keys, and values

Start with one embedding vector for each token.



$$q_i = W_Q^T x_i$$

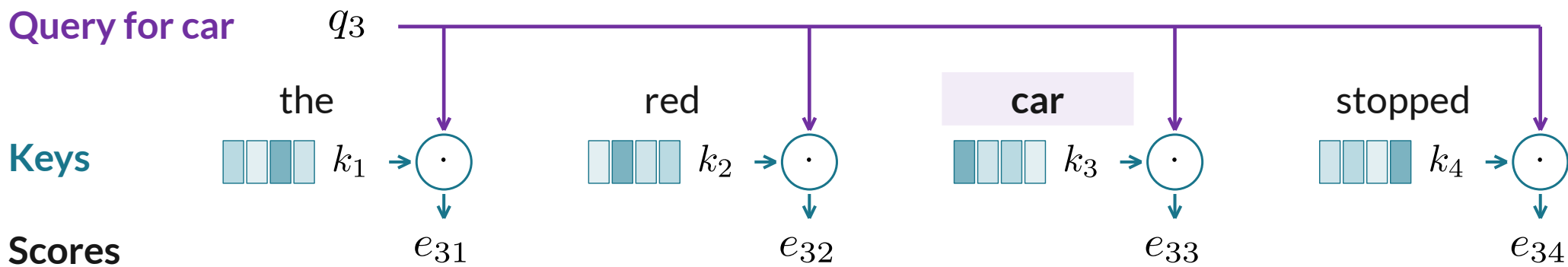
$$k_i = W_K^T x_i$$

$$v_i = W_V^T x_i$$

i : token position. The learned W matrices are shared across positions.

One query, all keys

For car, compare its query with every key—including its own.



$$e_{3j} = \frac{q_3^T k_j}{\sqrt{d_k}}$$

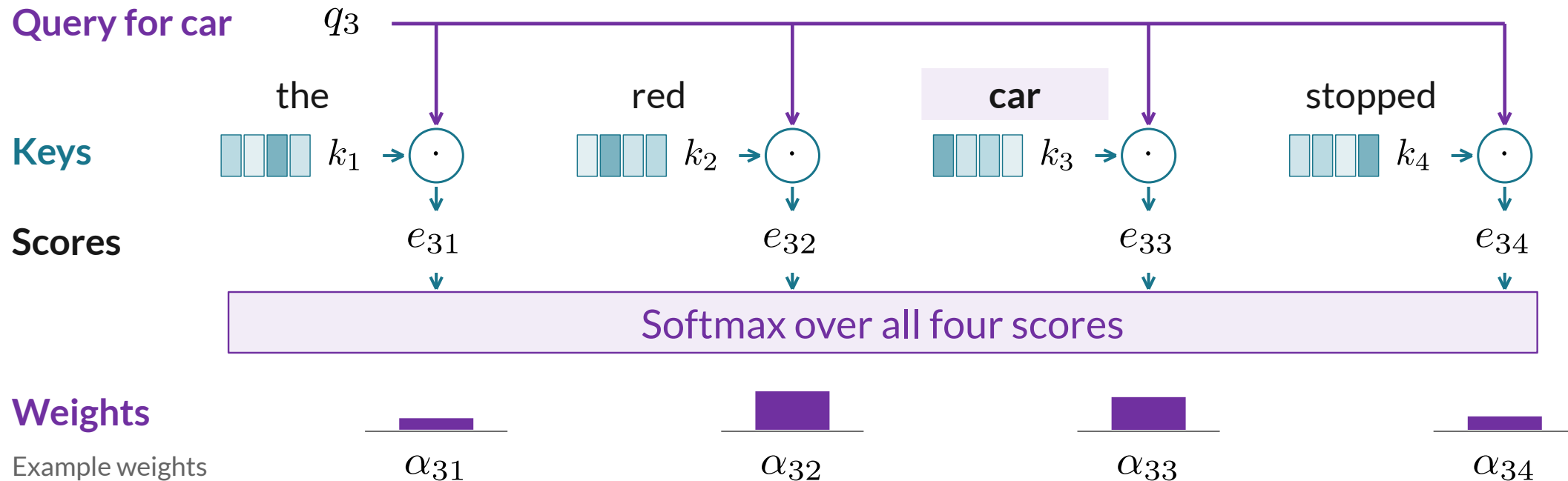
j: input position (1–4)

d_k : query/key vector length

A larger score means a stronger match to this query.

Attention weights

Normalize over input positions, not vocabulary words.

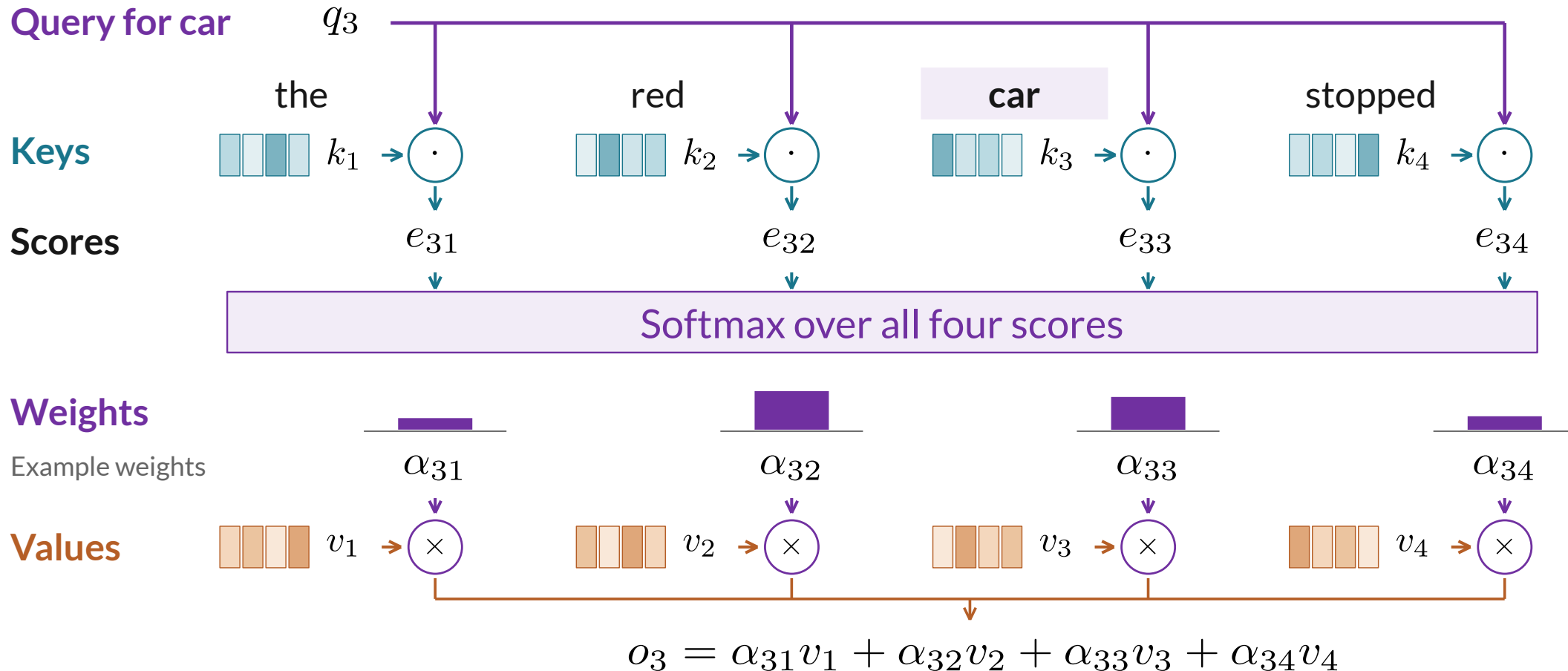


$$\alpha_{3j} = \frac{\exp(e_{3j})}{\sum_{r=1}^4 \exp(e_{3r})}$$

Nonnegative weights that sum to 1 over the four input positions.

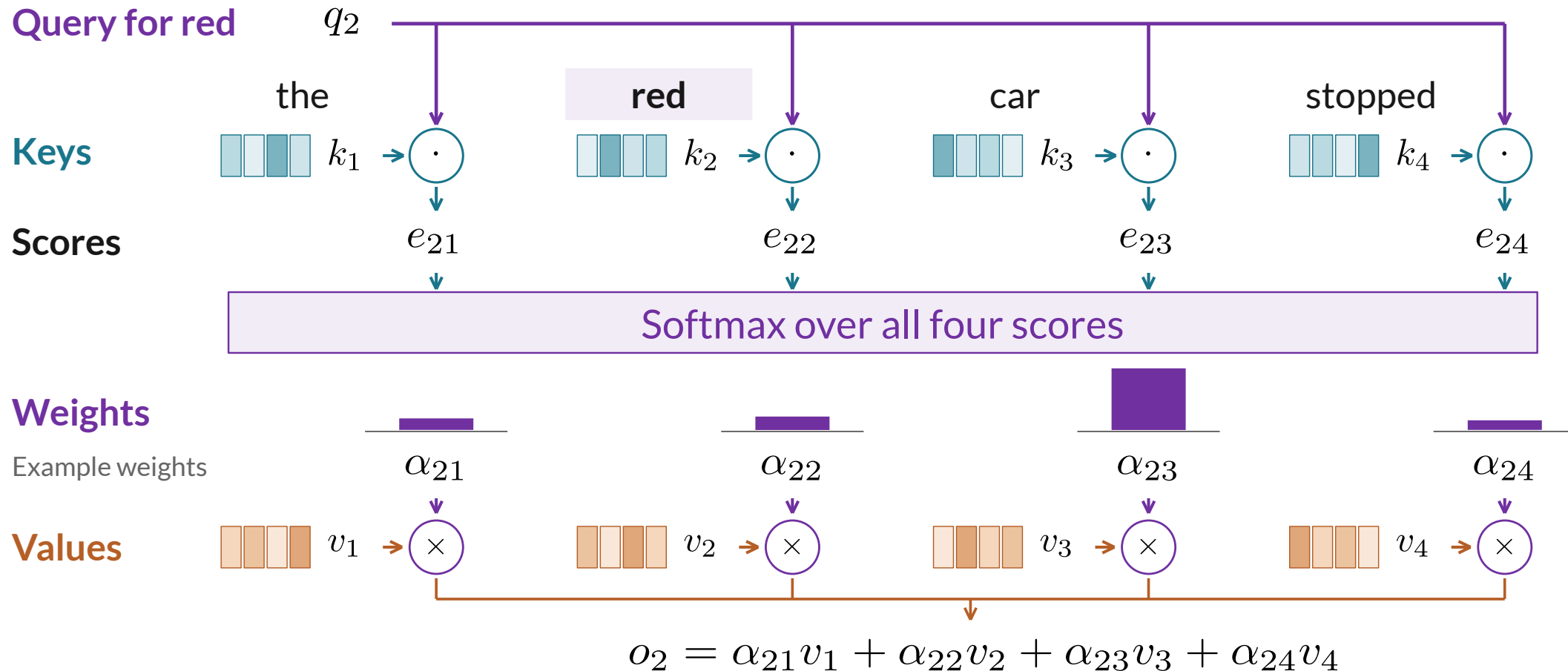
Weighted sum of values

The output is a feature vector, not a word prediction.



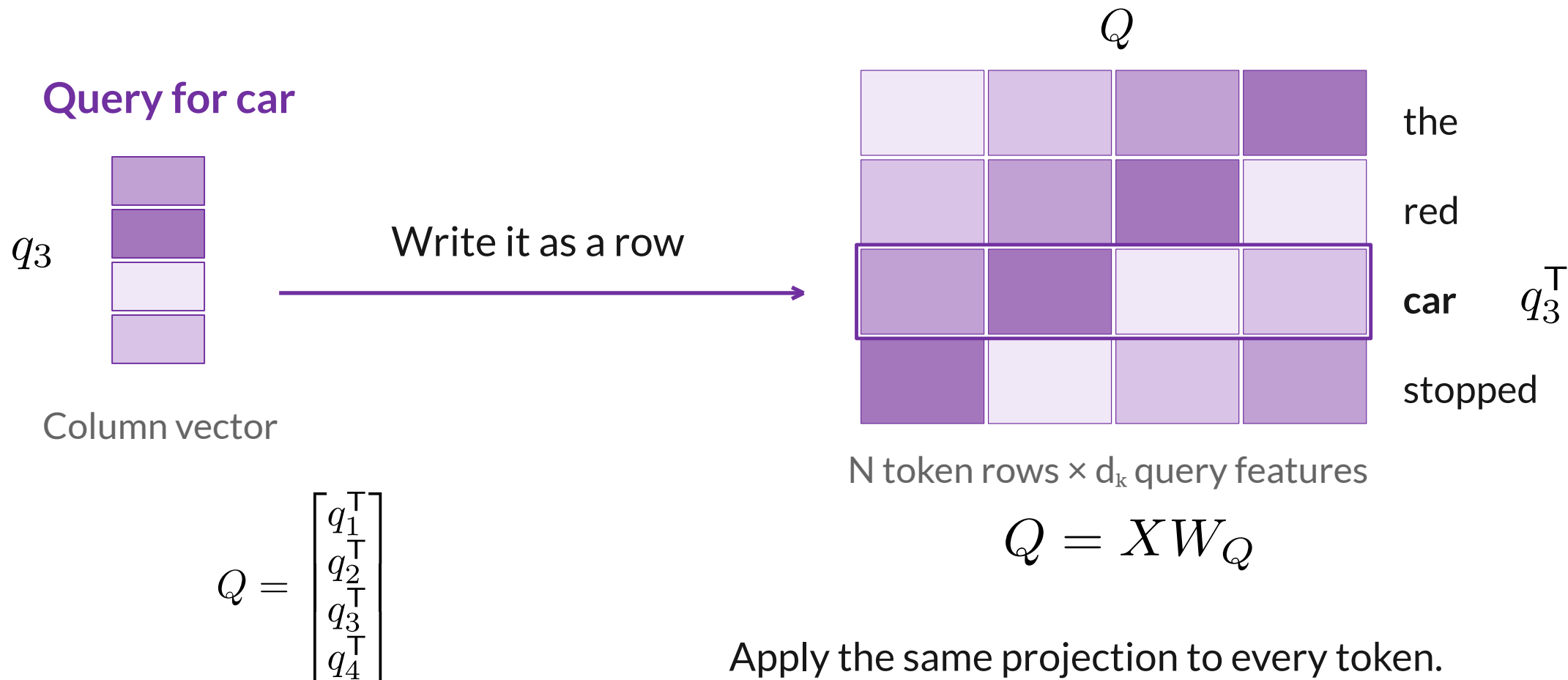
Another query, another output

For red, change the query. Keep the same keys and values.



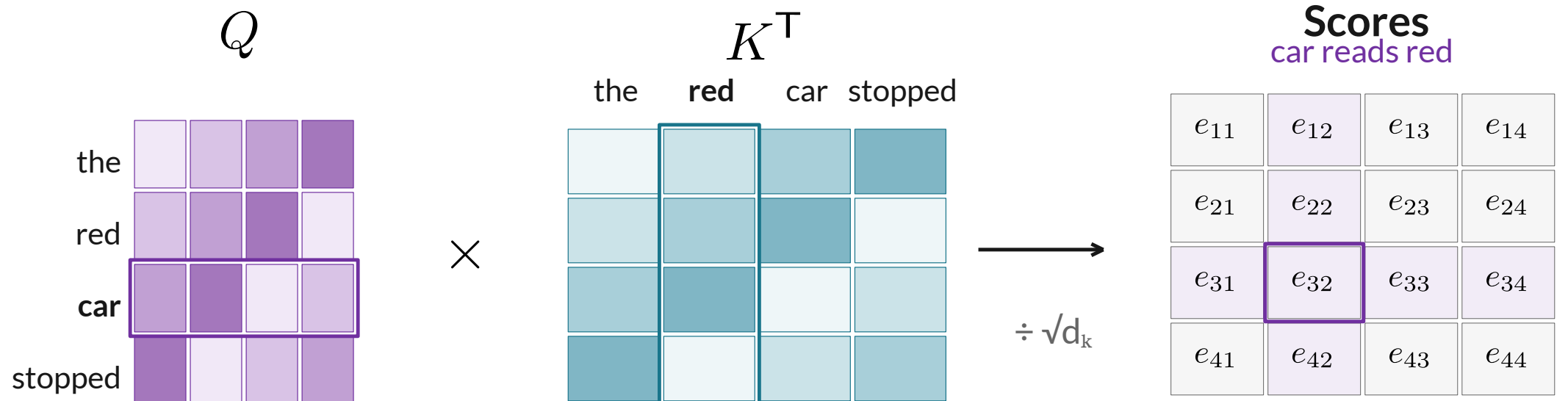
Vectors and matrix rows

Matrix formulas use one row for each token.



All pairwise scores

Each query is compared with every key.



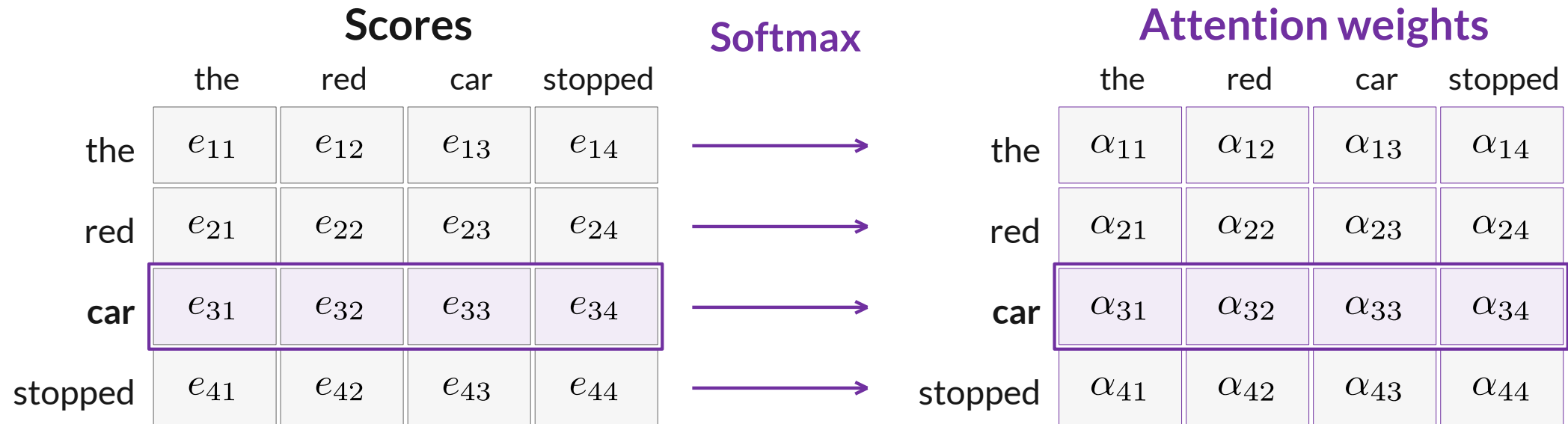
Rows: queries Columns: keys

$$E = QK^T / \sqrt{d_k}$$

$$e_{32} = \frac{q_3^T k_2}{\sqrt{d_k}}$$

Attention weights by row

Normalize each query's scores over all keys.



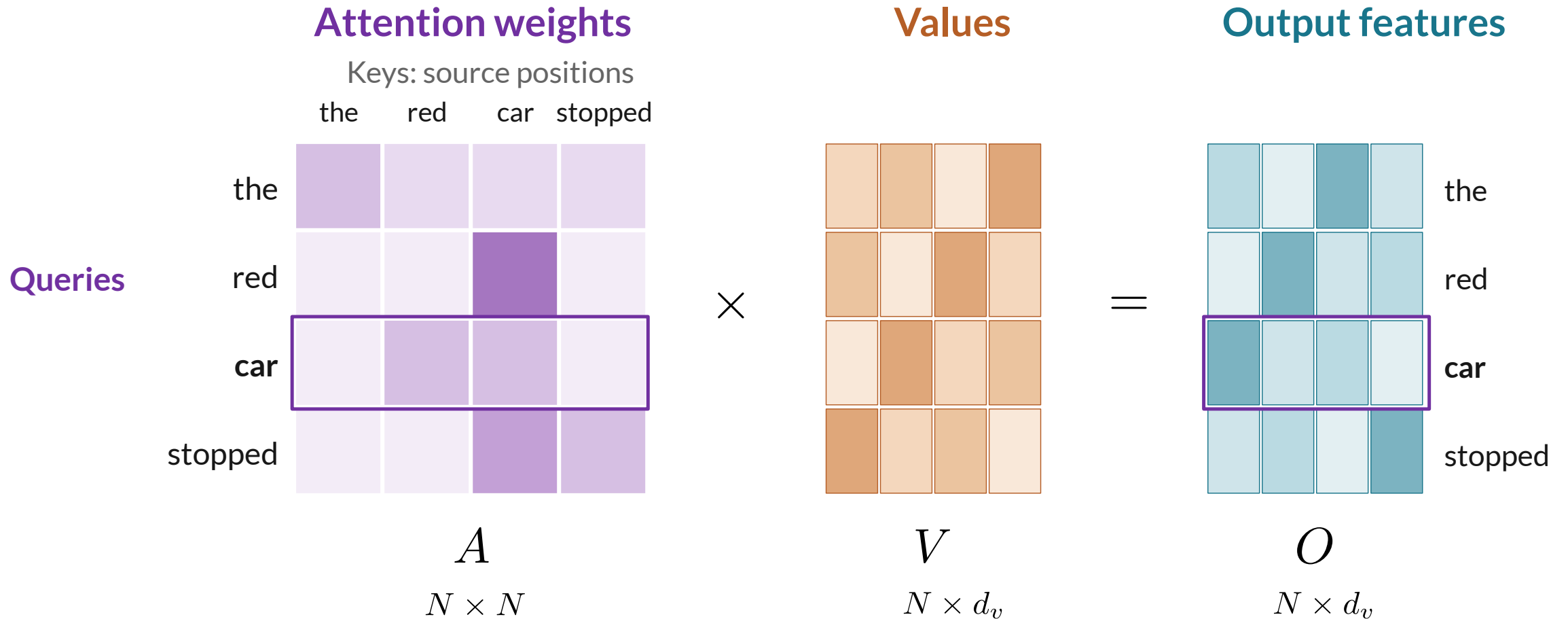
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{r=1}^N \exp(e_{ir})}$$

$$\sum_{j=1}^N \alpha_{ij} = 1$$

One distribution over input positions for each query.

Self-attention over the sequence

Each output row is a mixture of value vectors.



Self-attention computation

Input: N token vectors, each with d features

Project each token

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

$$Q, K : N \times d_k \quad V : N \times d_v$$

Compare every pair

$$E = QK^T / \sqrt{d_k}$$

$$E : N \times N$$

Normalize each row

$$A = \text{softmax}_{\text{row}}(E)$$

$$A : N \times N$$

Mix value vectors

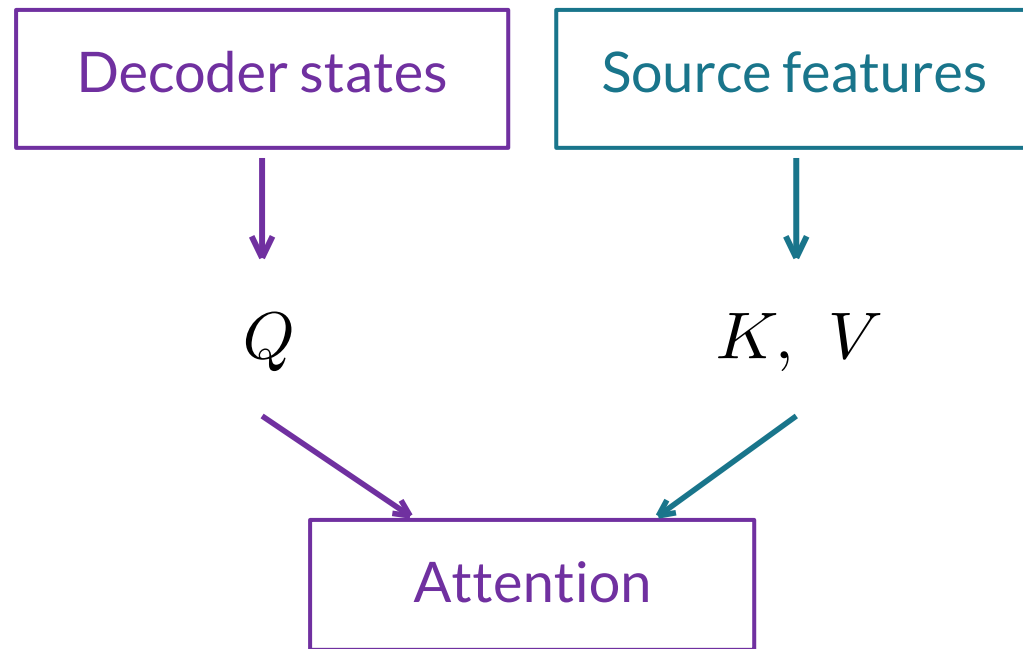
$$O = AV$$

$$O : N \times d_v$$

d_k : query/key width d_v : value/output width

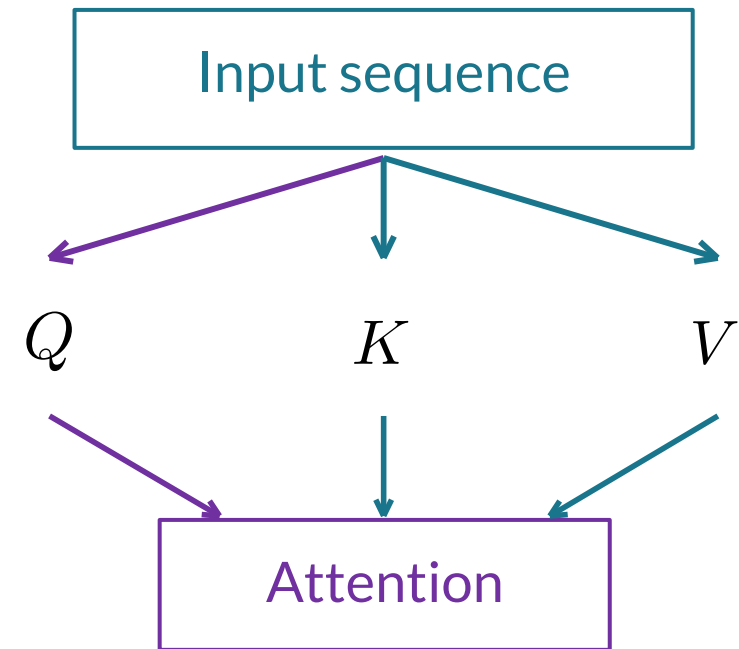
Cross-attention and self-attention

Cross-attention



Read a separate source

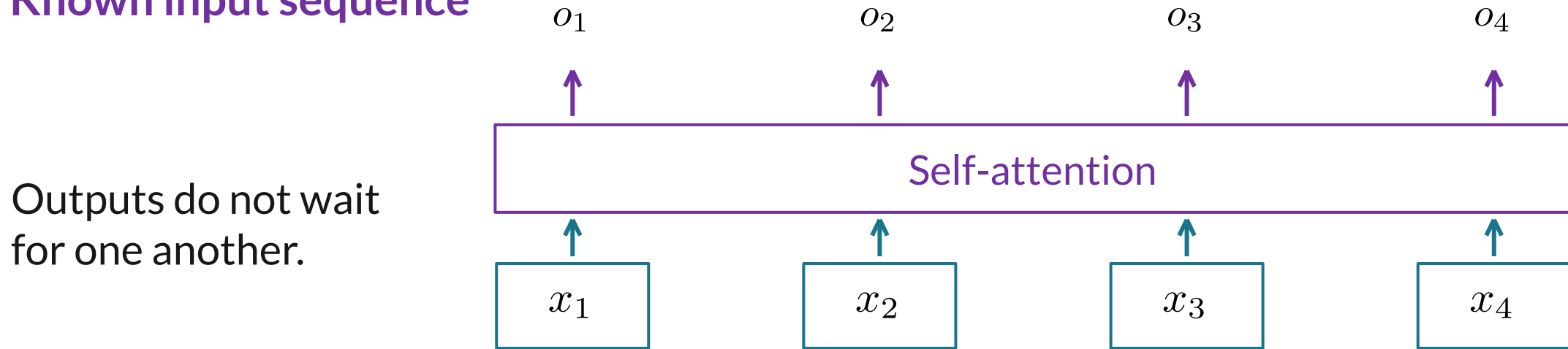
Self-attention



Build context within a sequence

Parallel sequence processing

Known input sequence



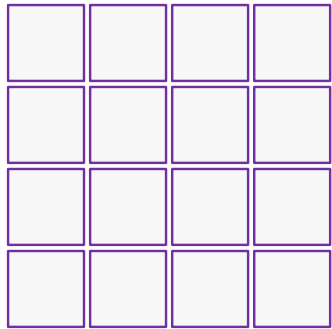
Autoregressive generation



Generation still waits for the selected next token.

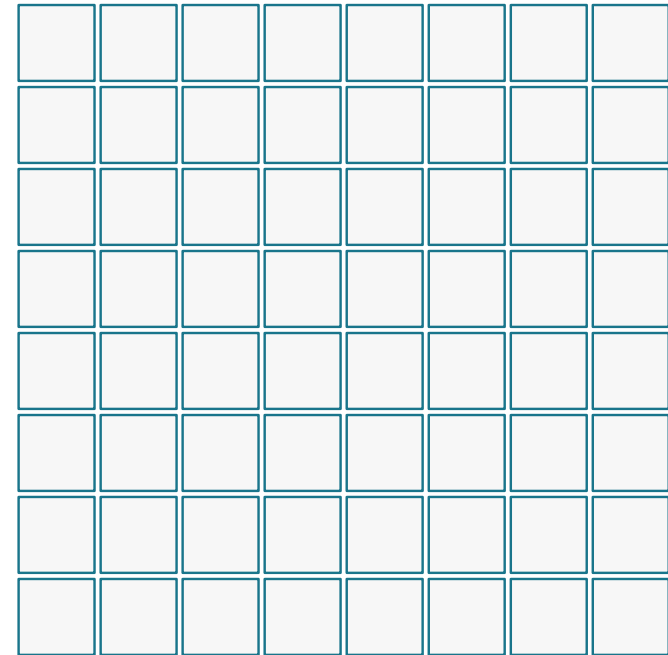
Pairwise computation

N input positions



N^2 pairs

2N input positions



$4N^2$ pairs



Twice the rows
and columns

Full attention compares every query with every key.

Word order

The same words can describe different events.

dog

does the biting

bites

man

gets bitten

man

does the biting

bites

dog

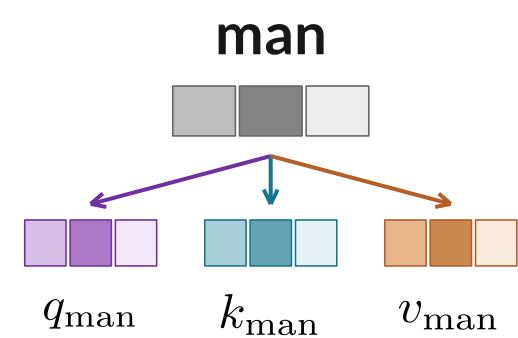
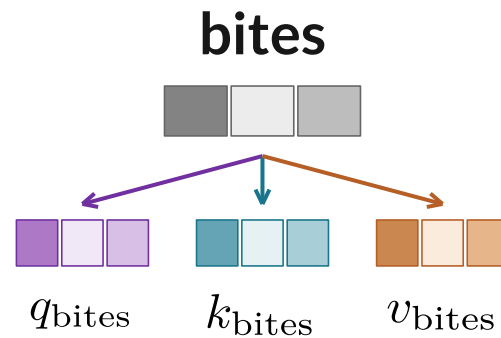
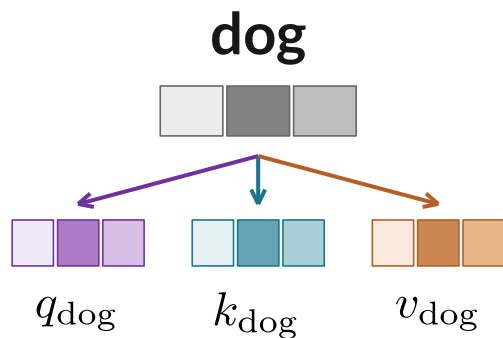
gets bitten

Content matching alone has no signal telling it which word came first.

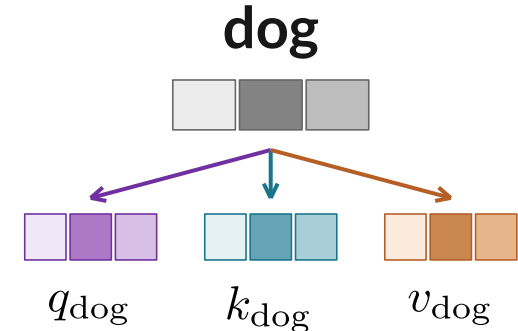
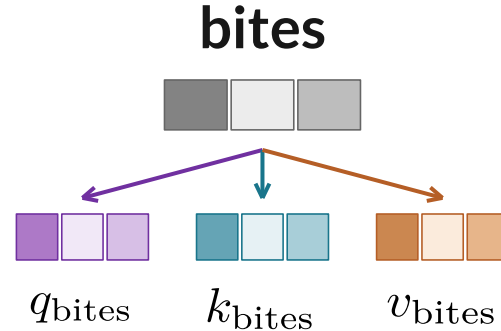
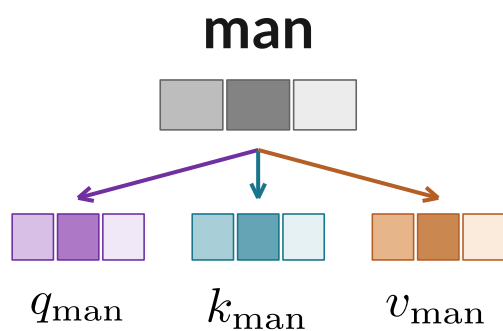
Reordering queries, keys, and values

No position information. Every query can read every key.

Original
order



Swapped
order



The same word uses the same learned projections at its new location.

Reordering attention weights

No position information. Every query can read every key.

Each query sees the same key vectors and scores after the swap.

	dog	bites	man
dog	<i>a</i>	<i>b</i>	<i>c</i>
bites			
man			

	man	bites	dog
man			
bites			
dog	<i>c</i>	<i>b</i>	<i>a</i>

Example weights

Rows: queries. Columns: keys. Both follow the new word order.

a, b, c track the weights of the dog query.

Reordering the weighted sum

No position information. Every query can read every key.

Track the output for dog.

$$a \times \begin{matrix} \text{dog} \\ \begin{array}{|c|c|c|} \hline \text{light} & \text{dark} & \text{light} \\ \hline \end{array} \\ v_{\text{dog}} \end{matrix} + b \times \begin{matrix} \text{bites} \\ \begin{array}{|c|c|c|} \hline \text{dark} & \text{light} & \text{light} \\ \hline \end{array} \\ v_{\text{bites}} \end{matrix} + c \times \begin{matrix} \text{man} \\ \begin{array}{|c|c|c|} \hline \text{light} & \text{dark} & \text{light} \\ \hline \end{array} \\ v_{\text{man}} \end{matrix} = \begin{matrix} \begin{array}{|c|c|c|} \hline \text{dark} & \text{light} & \text{light} \\ \hline \end{array} \\ o_{\text{dog}} \end{matrix}$$

$$c \times \begin{matrix} \text{man} \\ \begin{array}{|c|c|c|} \hline \text{light} & \text{dark} & \text{light} \\ \hline \end{array} \\ v_{\text{man}} \end{matrix} + b \times \begin{matrix} \text{bites} \\ \begin{array}{|c|c|c|} \hline \text{dark} & \text{light} & \text{light} \\ \hline \end{array} \\ v_{\text{bites}} \end{matrix} + a \times \begin{matrix} \text{dog} \\ \begin{array}{|c|c|c|} \hline \text{light} & \text{dark} & \text{light} \\ \hline \end{array} \\ v_{\text{dog}} \end{matrix} = \begin{matrix} \begin{array}{|c|c|c|} \hline \text{dark} & \text{light} & \text{light} \\ \hline \end{array} \\ o_{\text{dog}} \end{matrix}$$

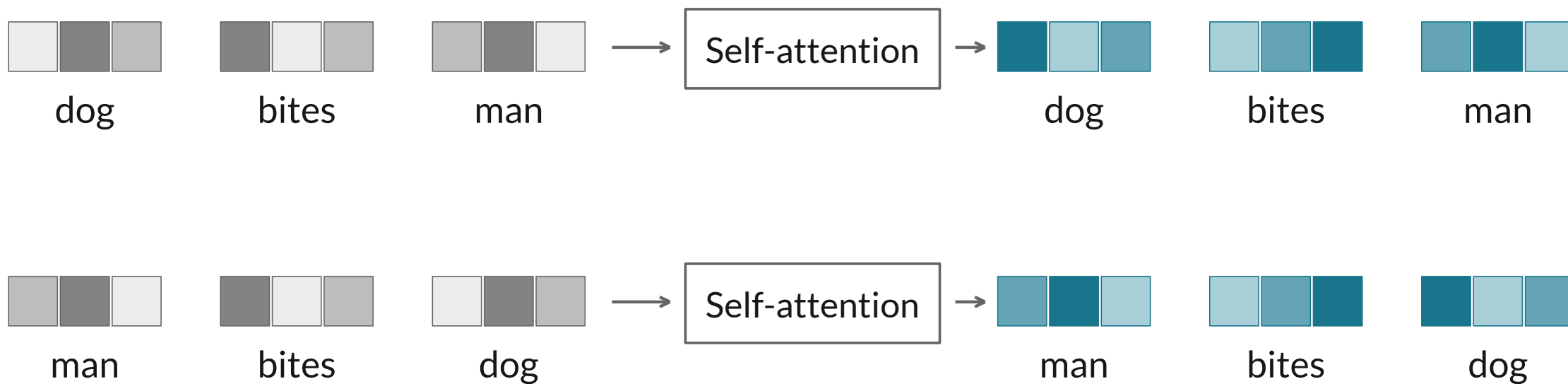
$$av_{\text{dog}} + bv_{\text{bites}} + cv_{\text{man}} = cv_{\text{man}} + bv_{\text{bites}} + av_{\text{dog}}$$

a, b, c: dog-query weights for dog, bites, man. v: value vector.

Permutation equivariance

No position information. Every query can read every key.

Input vectors



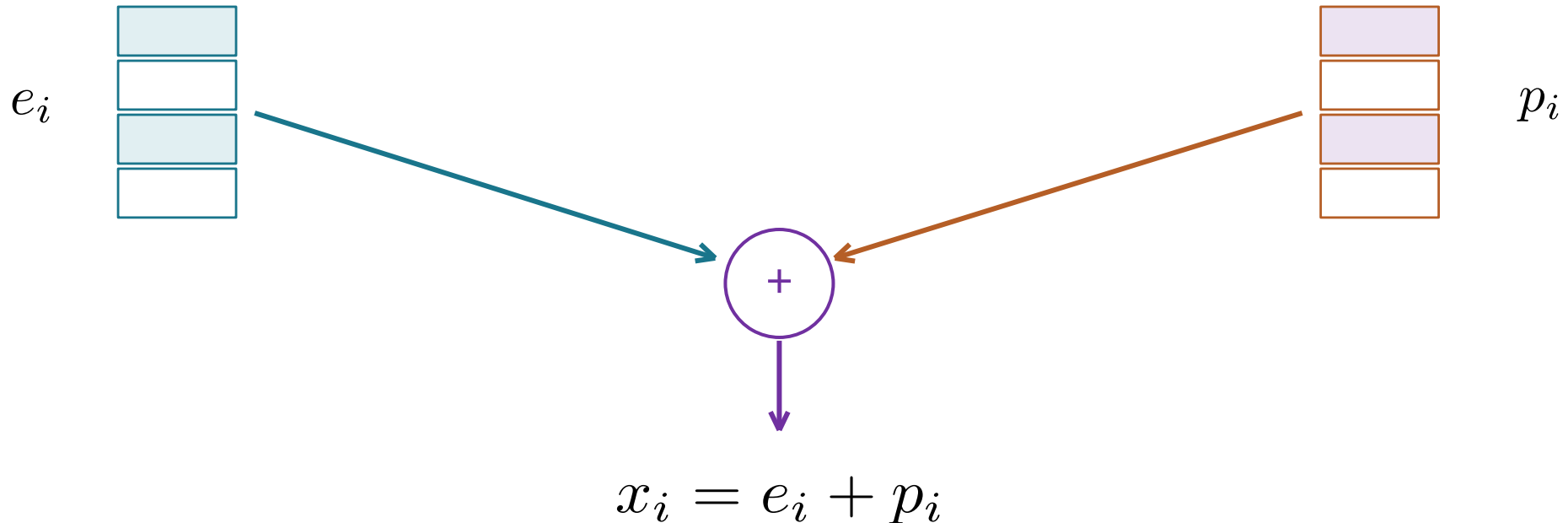
$$\text{SA}(\text{reorder}(X)) = \text{reorder}(\text{SA}(X))$$

Each output feature moves with its word. Both sides use the same rearrangement.

Content and position

Content embedding

Position embedding



i is the token's position. Use the combined features for attention.

Word order and position vectors

Position vectors stay attached to sequence locations.

Position 1

dog

$$e_{\text{dog}} + p_1$$

Position 2

bites

$$e_{\text{bites}} + p_2$$

Position 3

man

$$e_{\text{man}} + p_3$$

After swapping dog and man

man

$$e_{\text{man}} + p_1$$

bites

$$e_{\text{bites}} + p_2$$

dog

$$e_{\text{dog}} + p_3$$

$$e_{\text{dog}} + p_1 \neq e_{\text{dog}} + p_3$$

Next: Transformers

Attention

$$O = \text{softmax}_{\text{row-wise}} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

-
- Position information and causal masking
 - Multihead attention and feed-forward layers
 - Complete Transformer models for sequences and images