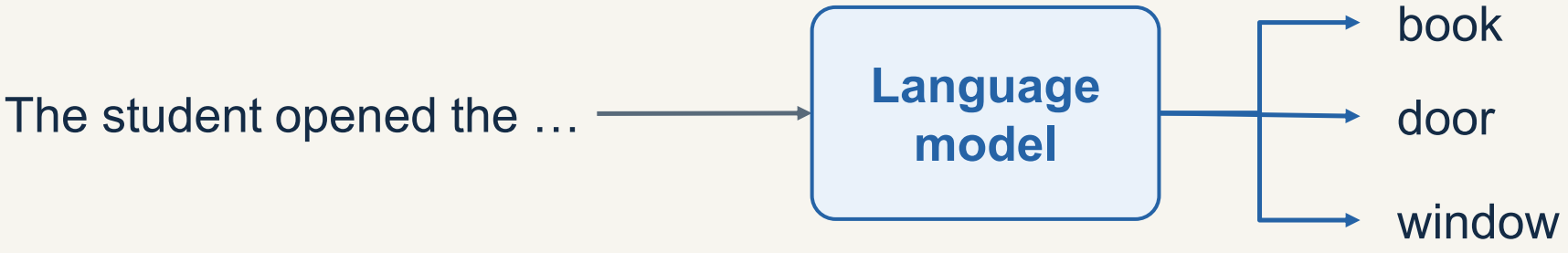
Introduction to NLP

Foundations for understanding modern language models

Andreas Makris | PhD student
Lancaster University, ProbAI Hub



NLP tasks and language models

TASK	INPUT		OUTPUT
Classification	I loved this book!	→	Positive
Translation	Good morning	→	Bonjour
Information extraction	Ada lives in London	→	Ada: person London: place
Question answering	What is the capital of France?	→	Paris
Text generation	The student opened the …	→	book

Language models can do most NLP tasks through language generation

What makes language difficult?

The trophy did not fit in the suitcase
because **it** was too ...

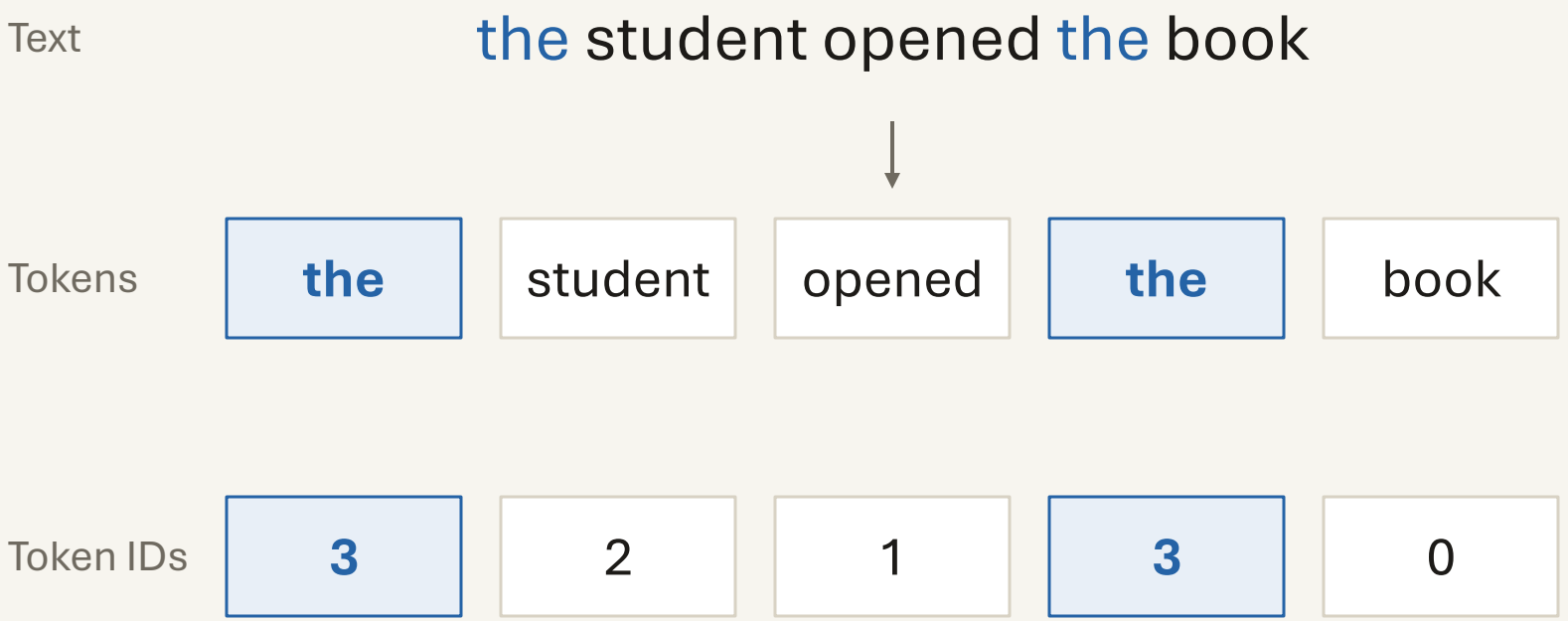
big



small



Word-level tokenization and vocabulary



VOCABULARY

Token	ID
book	0
opened	1
student	2
the	**3**

Sequence length: T = 5

Vocabulary size: V = 4

Downside: Treating token IDs as numerical features imposes arbitrary ordering and distances.

One-hot encoding and its limitations

Token	book	opened	student	the
the	0	0	0	1
student	0	0	1	0
opened	0	1	0	0
the	0	0	0	1
book	1	0	0	0

$$T \times V = 5 \times 4$$

Vectors grow with vocabulary size

Distinct tokens are equally distant. What about unseen words?

T: sequence length
V: vocabulary size
d: embedding dimension
X: one-hot matrix
E: embedding matrix
Z: sequence embeddings

Learned embeddings

X

	book	opened	student	the
the	0	0	0	1
student	0	0	1	0
opened	0	1	0	0
the	0	0	0	1
book	1	0	0	0

T × V = 5 × 4

×

E

book	0.7	0.1	−0.3
opened	−0.4	0.8	0.2
student	0.3	−0.2	0.6
the	**0.1**	**0.5**	**−0.7**

V × d = 4 × 3

=

Z

0.1	**0.5**	**−0.7**
0.3	−0.2	0.6
−0.4	0.8	0.2
0.1	**0.5**	**−0.7**
0.7	0.1	−0.3

T × d = 5 × 3

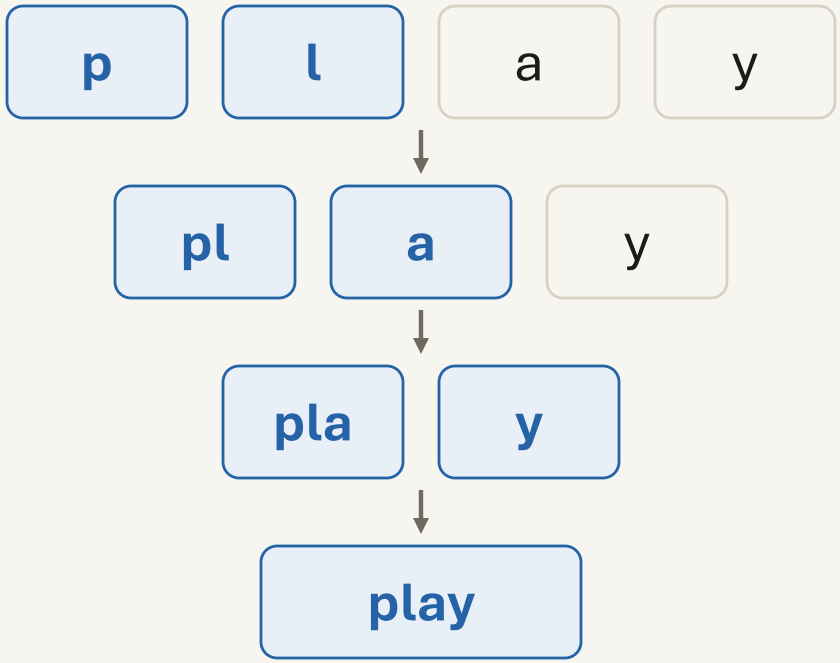
Multiplying by a one-hot vector selects a row of E

BPE: byte-pair encoding

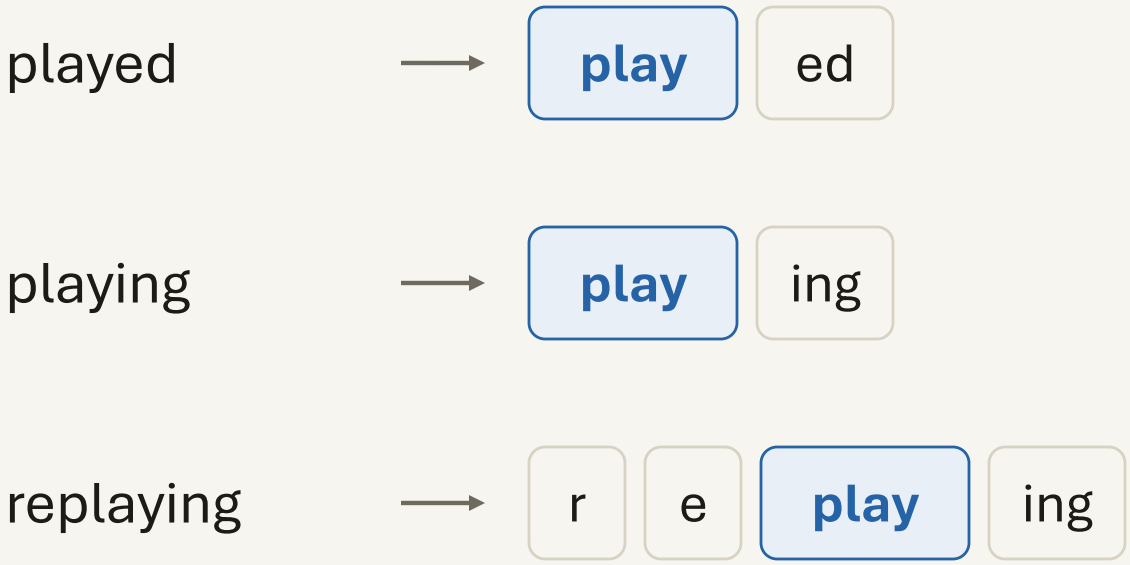
Subword tokenization and BPE

A smarter way to handle unknown words!

Merge frequent adjacent pairs



Reuse learned pieces



A fixed subword vocabulary can represent previously unseen words

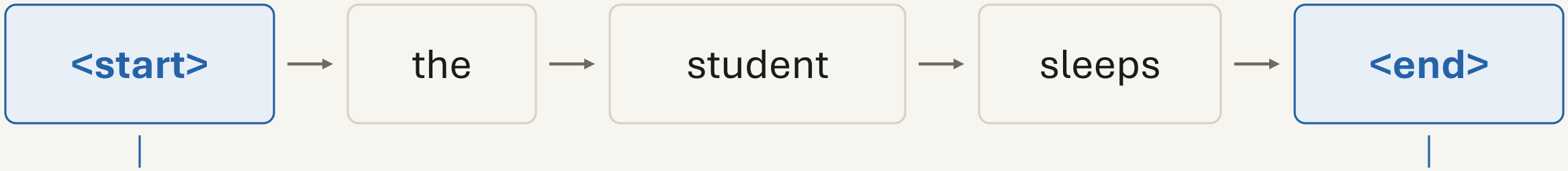
Sennrich et al. (2016)

<start> / BOS: beginning of sequence

<end> / EOS: end of sequence

Special tokens: sequence boundaries

A sequence with explicit boundaries



Begin the sequence

Signal completion

The model learns when to end,
just as it learns which token comes next.

Language modeling: predicting the next token

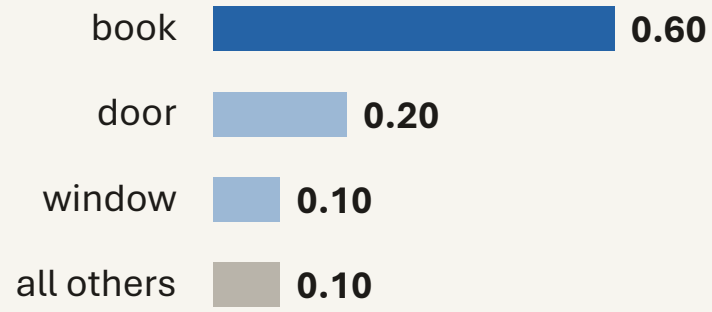
x_t: token at position t
$x_{<t}$: preceding tokens
T: sequence length
p_θ : model probabilities
θ: model parameters

Tokens so far

<start> the student opened the

→ **Language model** →

Next-token probabilities



One probability per vocabulary token

"All others" combines the remaining tokens.

Chain rule of probability

$$p_\theta(x_1, \ldots, x_T) = \prod_{t=1}^{T} p_\theta(x_t | x_{<t})$$

A language model assigns probabilities to token sequences

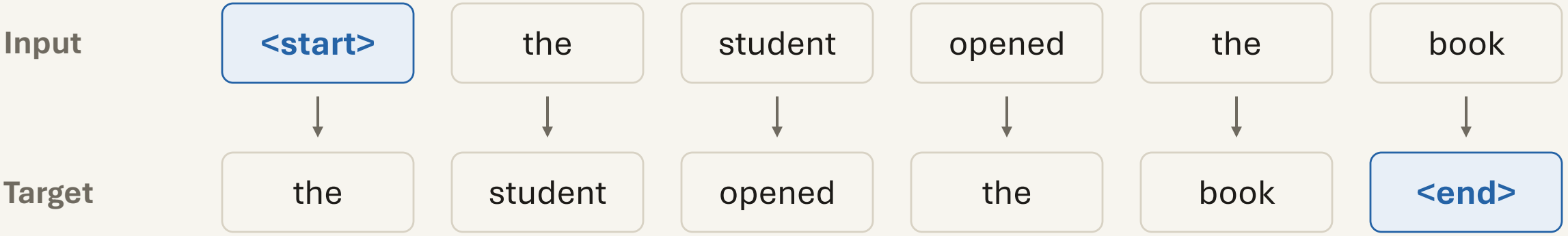
Training a language model

T: prediction positions
V: vocabulary size
$\mathcal{L}$: sequence loss

x_t: target token at t
$x_{<t}$: preceding tokens
p_θ : model probabilities
θ: model parameters

Teacher forcing

During training, the context contains the true preceding tokens.



Predict the next token using only the preceding tokens. This is **self-supervised learning**: the text itself supplies the next-token targets.

Cross-entropy / negative log-likelihood

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p_\theta(x_t | x_{<t})$$

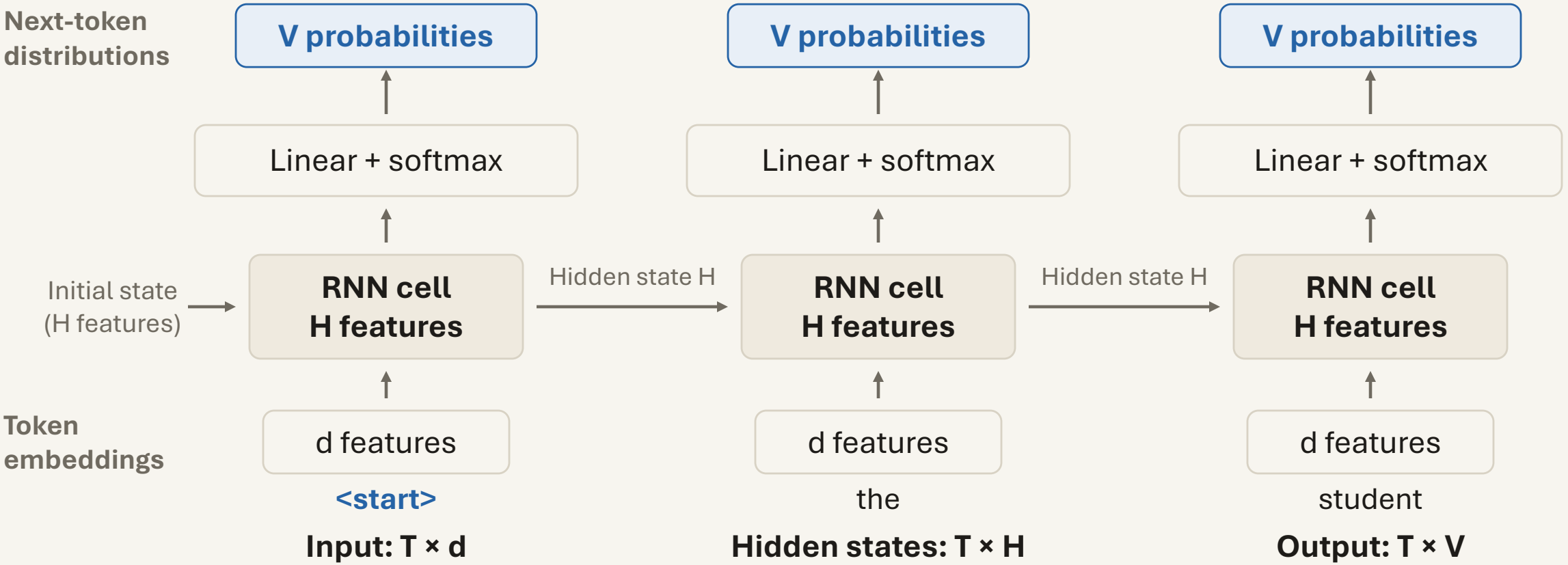
Model output after softmax

T × V probabilities

The text provides its own training targets

T: sequence length
d: embedding dimension
H: hidden-state dimension
V: vocabulary size

Recurrent neural networks (RNNs)



The same parameters are reused at every step.

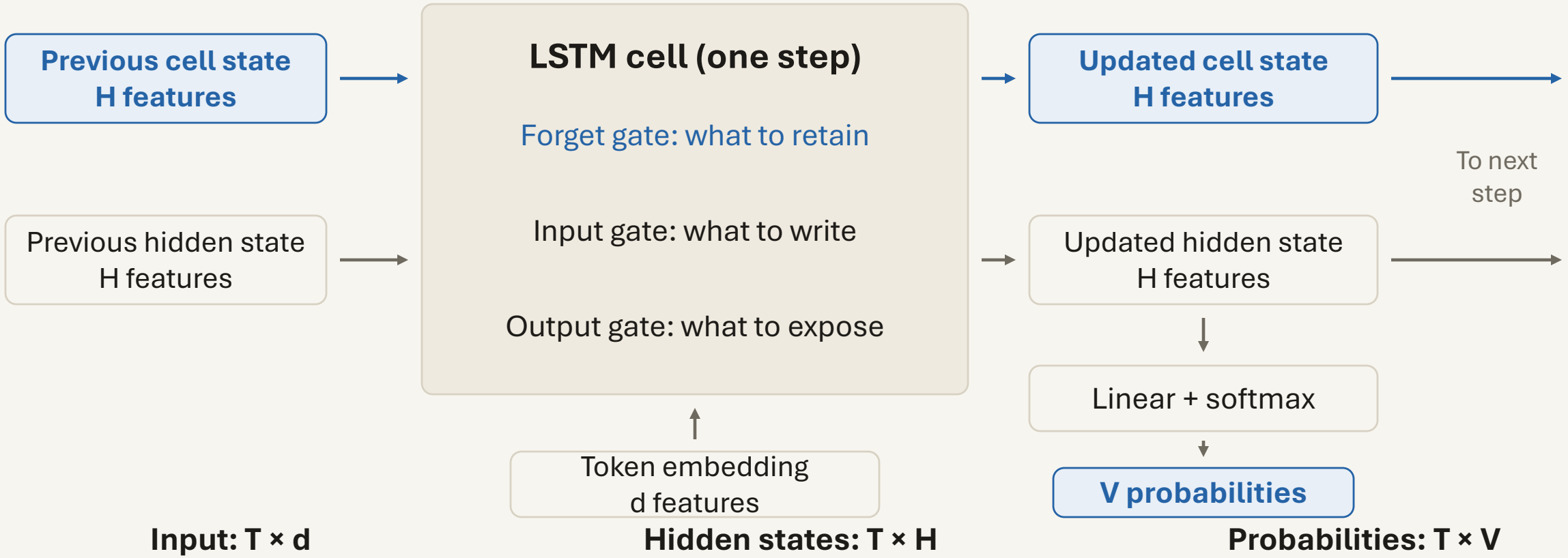
A hidden state carries information from earlier tokens

Long-range dependencies can be difficult to learn.

Hochreiter & Schmidhuber (1997)

Long short-term memory (LSTM)

T: sequence length
d: embedding dimension
H: hidden/cell-state dimension
V: vocabulary size



A separate cell state helps preserve information over time

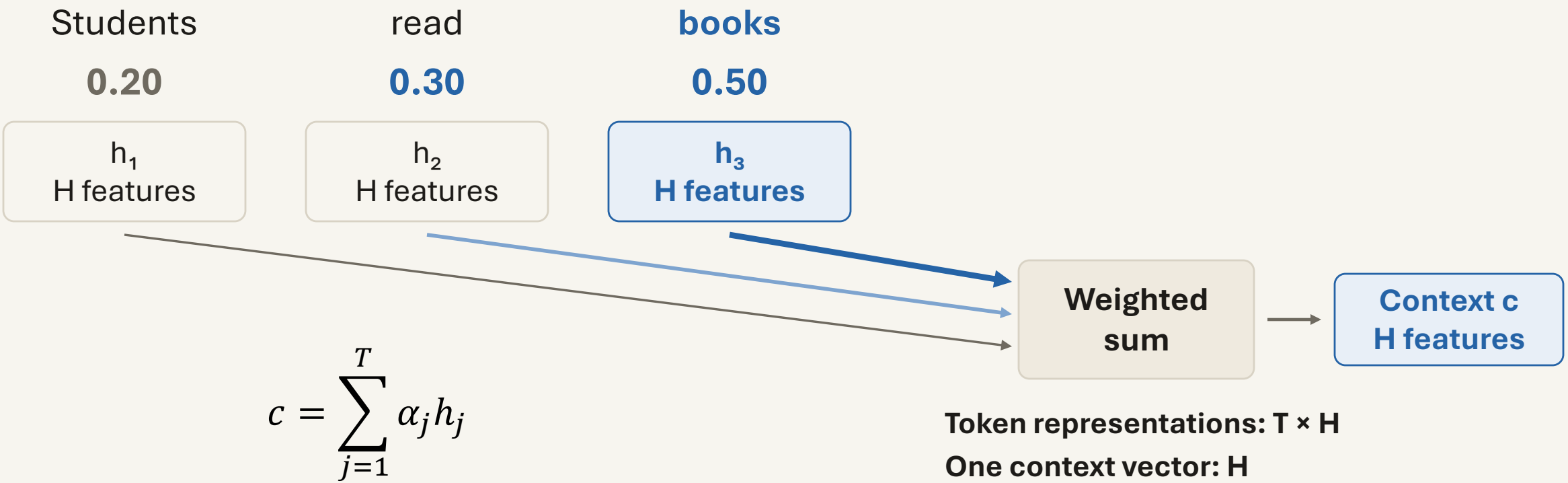
Recurrence still requires sequential computation.

PyTorch LSTM documentation

T: available tokens
H: representation dimension
h_j: representation at position j
α_j: weight at position j
c: context vector

Attention: a weighted view of context

Computing context for books



$$c = \sum_{j=1}^{T} \alpha_j h_j$$

Token representations: T × H

One context vector: H

Attention combines token representations using context-dependent weights

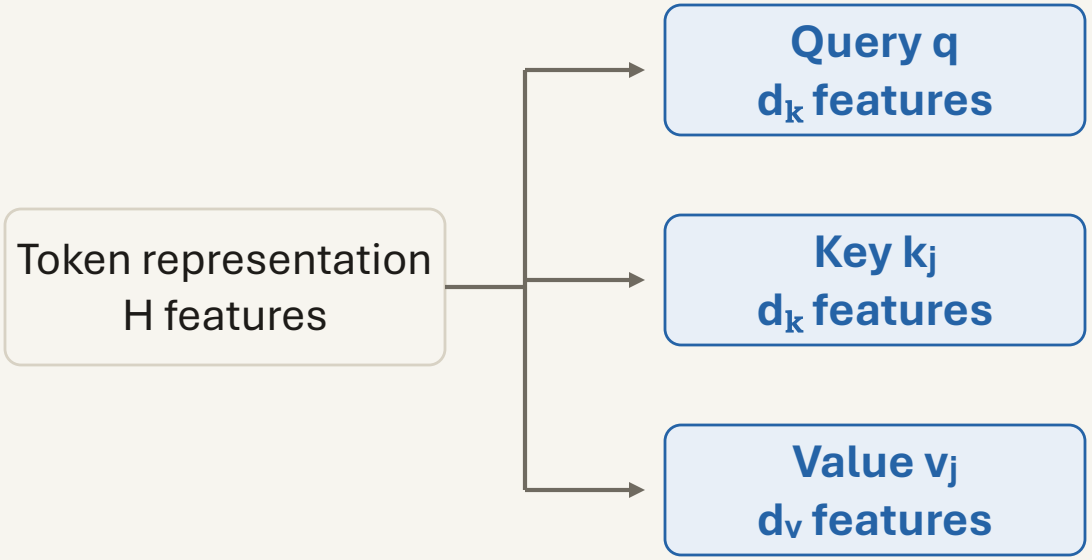
Bahdanau et al. (2015) Vaswani et al. (2017)

Queries, keys, and values

H: input dimension
d_k: query/key dimension
d_v: value/output dimension
T: transpose
s: score vector
q: current query
k_j: key at token j
v_j: value at token j
s_j: score at token j
c: attention output

Vaswani et al. (2017)

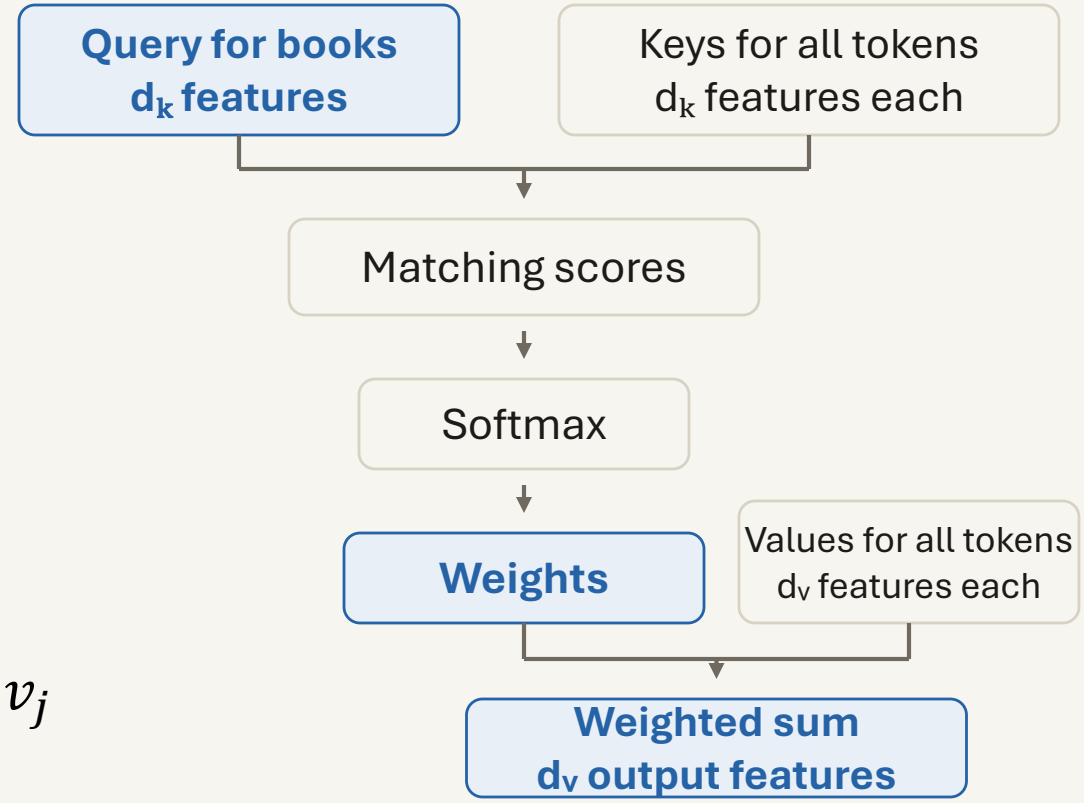
Learned linear projections



Apply to every token

$$s_j = \frac{q^\top k_j}{\sqrt{d_k}} \qquad c = \sum_j \text{softmax}(\mathbf{s})_j v_j$$

Attention for books

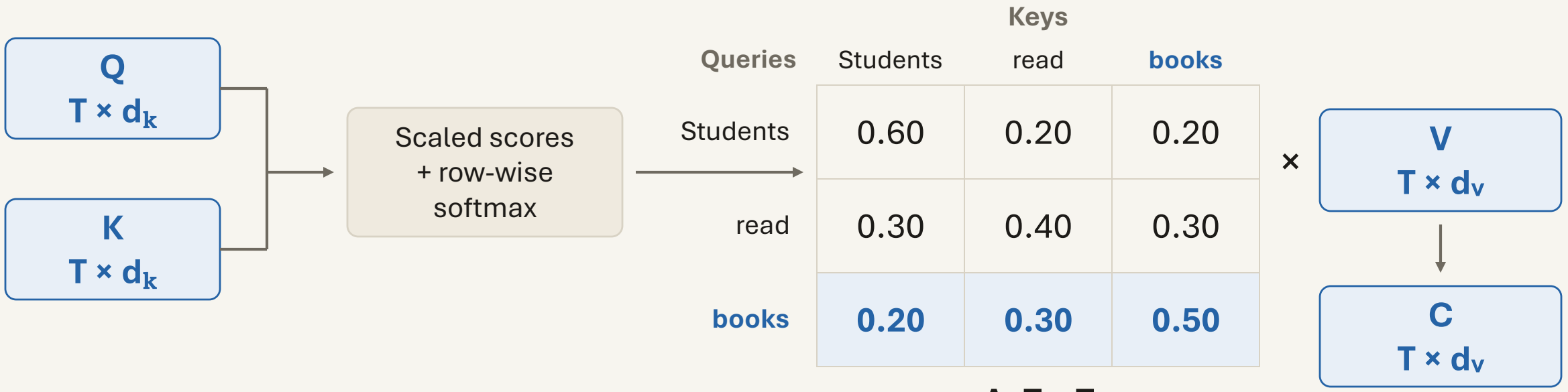


Queries and keys determine the weights used to combine values

Self-attention in matrix form

T: sequence length
d_k: query/key dimension
d_v: value/output dimension
T: transpose

Q: query matrix
K: key matrix
V: value matrix
A: attention weights
C: output representations



$$\mathbf{C} = \text{softmax}_{\text{row}}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}$$

Each row computes a weighted combination of the value vectors

Vaswani et al. (2017)

Causal self-attention

T: sequence length
d_v: value/output dimension
S: scaled matching scores
M: causal mask
A: attention weights

Causal mask M

	Students	read	books
Students	0	−∞	−∞
read	0	0	−∞
books	0	0	0

Mask: T × T

Add mask
to scores,
then apply softmax

Attention weights A

	Students	read	books
Students	1.00	0.00	0.00
read	0.43	0.57	0.00
books	0.20	0.30	0.50

Weights: T × T

$$\mathbf{A} = \mathrm{softmax}_{\mathrm{row}}(\mathbf{S} + \mathbf{M})$$

Output representations:
T × d_v

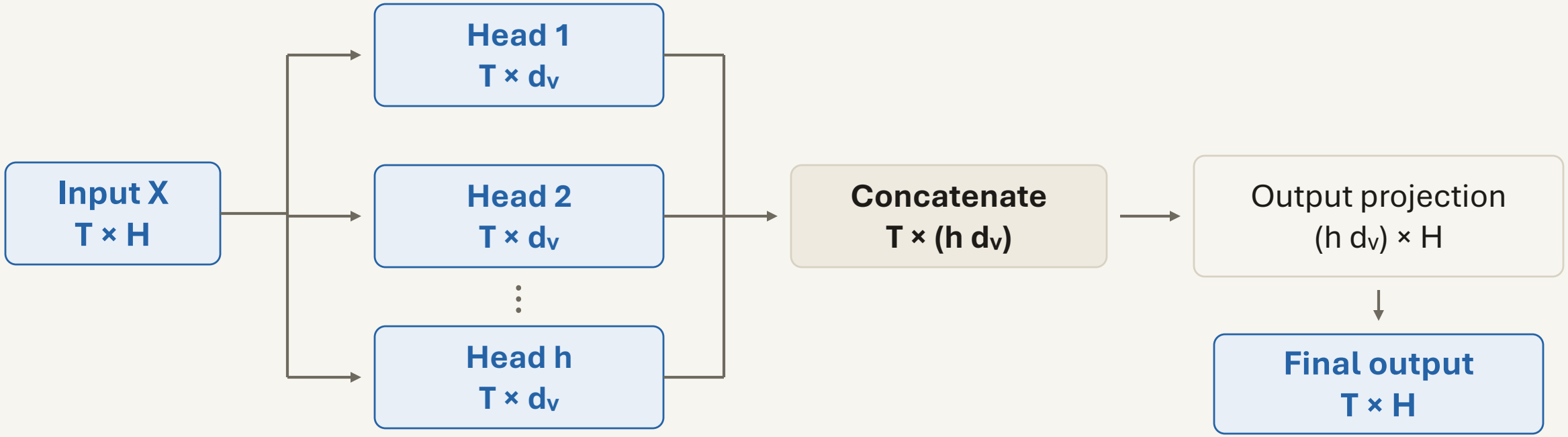
Each position can attend to itself and earlier positions

Vaswani et al. (2017)

Multi-head self-attention

X: input representations
T: sequence length
H: model dimension
h: number of heads

d_v: output width per head
W_O : output projection
$head_i$: output of head i



Each head learns its own query, key, and value projections.

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_\text{h})W_O$$

Each position can combine information through several attention patterns

Vaswani et al. (2017)

T: sequence length
H: model dimension
X: combined input

X_{tok} : token embeddings
P: positional embeddings

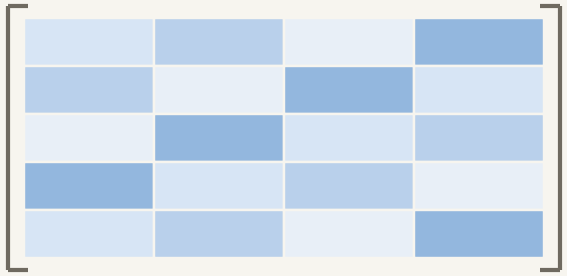
Where does token order come from?

Same tokens, different order

the	**dog**	bites	the	**man**

the	**man**	bites	the	**dog**

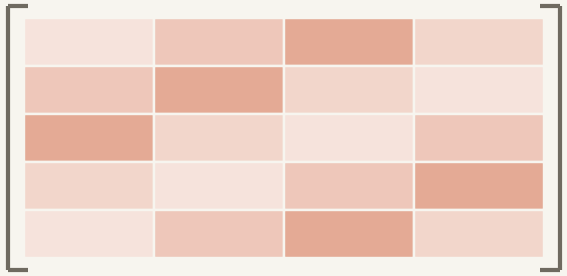
Token embeddings



T × H

+

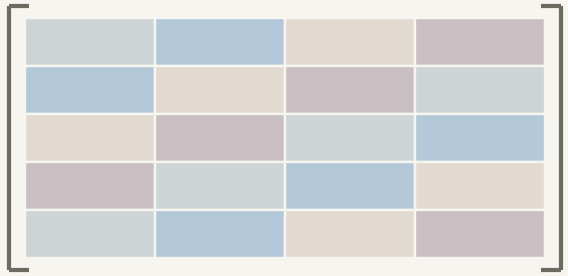
Positional embeddings



T × H

→

Transformer input



T × H

$$X = X_{\text{tok}} + P$$

Representations need information about both content and position

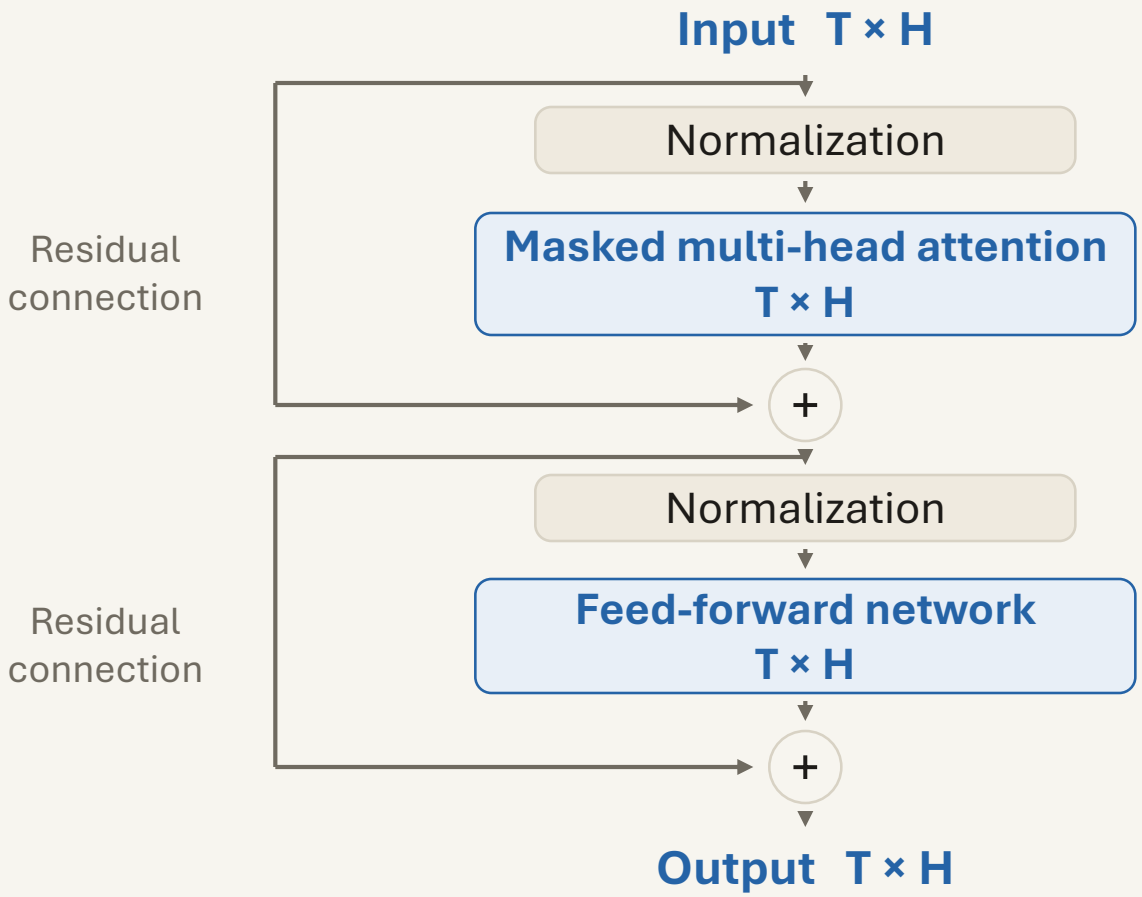
Vaswani et al. (2017)

T: sequence length

H: model dimension

F: feed-forward hidden dimension

A Transformer decoder block

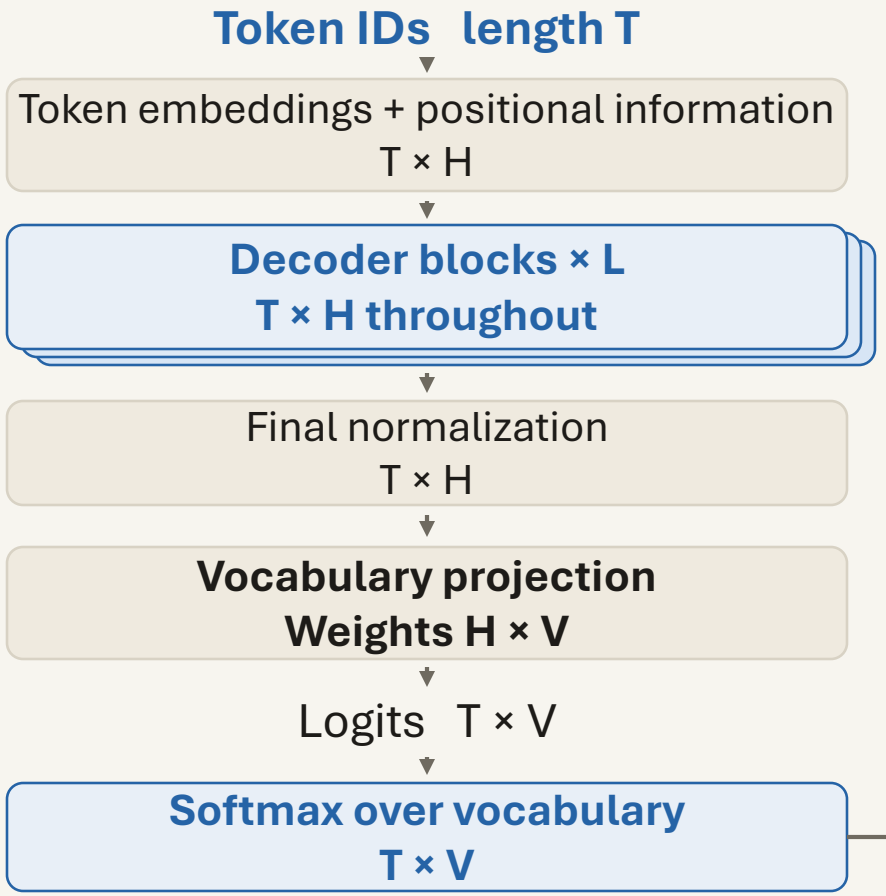


Attention "mixes" tokens. The feed-forward network transforms each token.

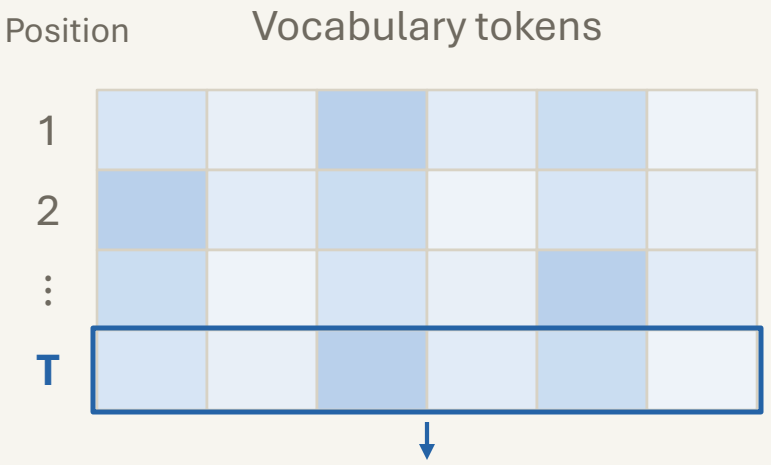
Xiong et al. (2020)

T: sequence length
H: model dimension
L: number of decoder blocks
V: vocabulary size

A Transformer language model



Next-token probabilities



Use the final row to predict the next token

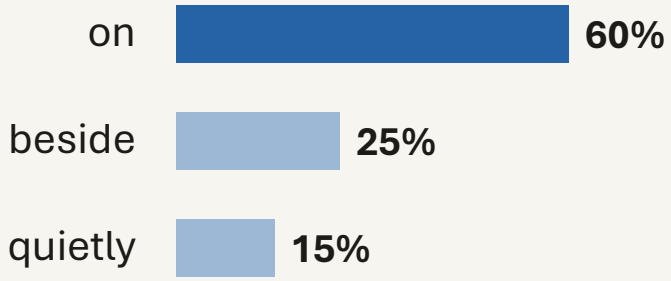
Each position produces a probability distribution over the vocabulary.

Vaswani et al. (2017)

z: vector of next-token logits
p: vector of next-token probabilities
τ: temperature, greater than zero

Choosing the next token

The cat sat ___



Greedy decoding
Choose the highest probability
on

Sampling
Draw from the distribution
One draw
quietly

Lower temperature
More concentrated



Higher temperature
More spread out



Temperature

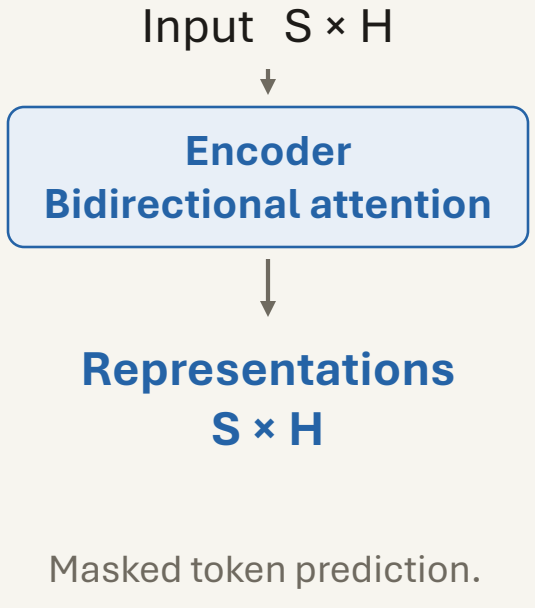
$$p = \text{softmax}\left(\frac{z}{\tau}\right)$$

The model provides probabilities. The decoding rule chooses the token.

Three Transformer architectures

S: source/input sequence length
T: decoder prefix length
H: model dimension
V: vocabulary size

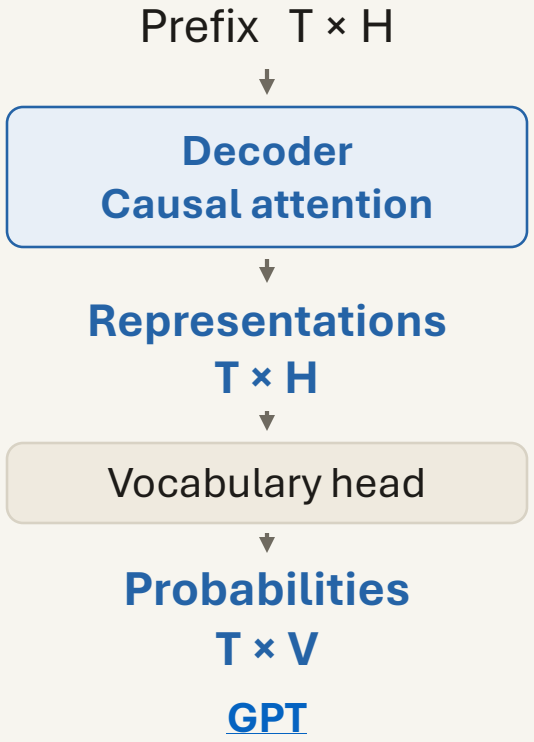
Encoder



Masked token prediction.

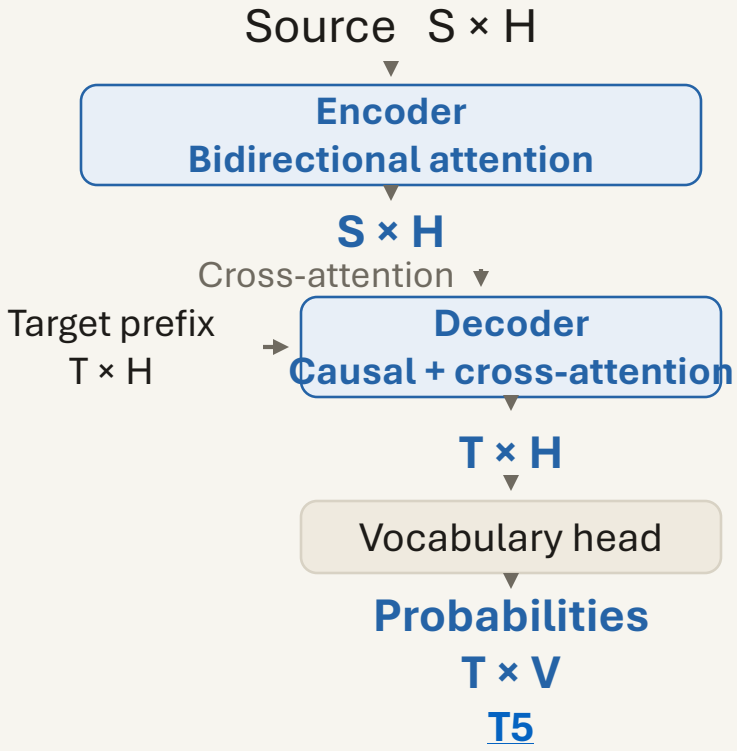
BERT

Decoder



GPT

Encoder–decoder

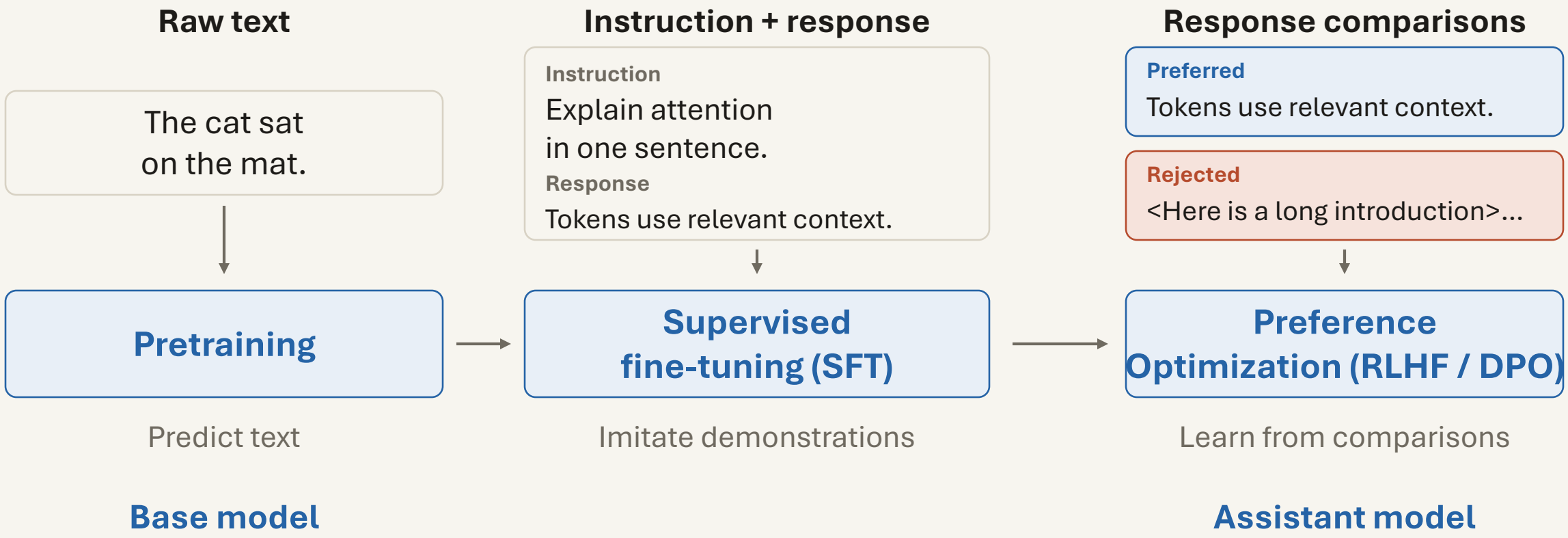


T5

The architecture determines which tokens can exchange information.

Training an LLM assistant

A common training pipeline



The architecture can stay the same while the data and training objective change.

Ouyang et al. (2022) Rafailov et al. (2023)

Learning from response preferences

Explain attention in one sentence.

A Preferred

Tokens use relevant context.

B Rejected

<Here is a long introduction>...

Preference comparisons

RLHF

Reinforcement learning from human feedback

Reward model

Prompt + response
→ one scalar score

→

Reinforcement learning

Update language model

DPO

Direct preference optimization

Directly update language model

Using preferred/rejected pairs

Comparisons provide a training signal about which responses people prefer.

Ouyang et al. (2022) Rafailov et al. (2023)

Prompting and in-context learning

One prompt

Classify sentiment as positive or negative.

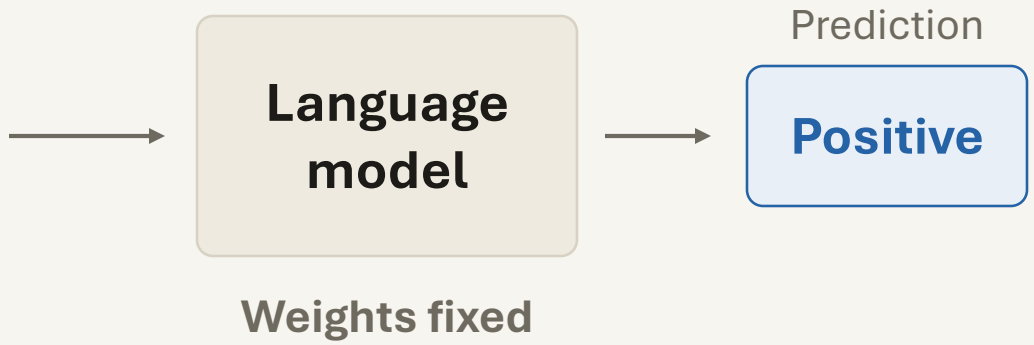
Examples

Review: “Loved it.”
Sentiment: Positive

Review: “Very disappointing.”
Sentiment: Negative

New input

Review: “A wonderful experience.”
Sentiment: ___



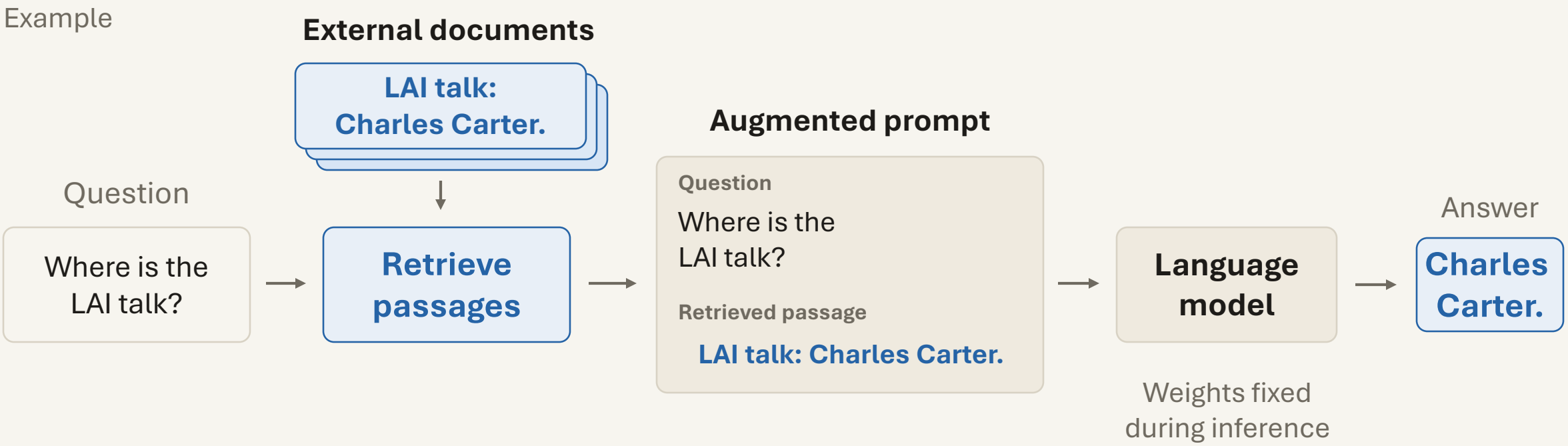
Zero-shot / one-shot: no / one example(s) in the prompt

Few-shot: a few examples in the prompt

Examples guide the response through context. The model’s weights stay fixed.

Brown et al. (2020)

Retrieval-augmented generation (RAG)



Retrieved documents become part of the model's input.

Lewis et al. (2020)

PPL: perplexity

ℓ: mean negative log-likelihood per token, using natural logs

Evaluating language models

Evaluate on held-out data

Next-token prediction

the cat ___

↓

Observed: sat

Loss / perplexity (lower is better, lower bound is 1)

$$\mathrm{PPL} = \exp(\ell)$$

Task performance

Question: 2 + 2?

Prediction 4

Reference 4

Match

Accuracy / task-specific metrics

Response quality

Same prompt

Response 1: Clear answer

Response 2: Long digression

Compare

Human or model judgements

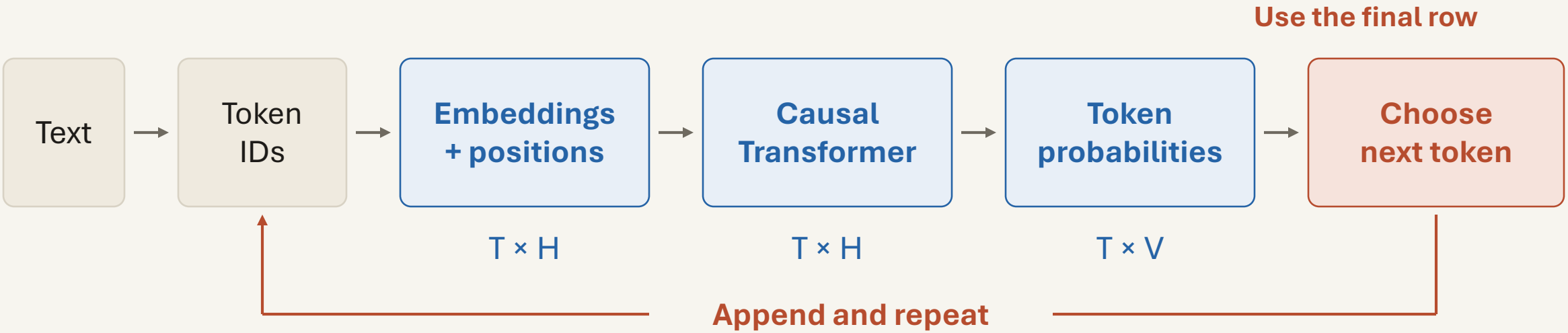
Different metrics measure different abilities.

Perplexity: Hugging Face

Liang et al. (2023)

T: current sequence length
H: model dimension
V: vocabulary size

A language model in one diagram



Training: update the weights

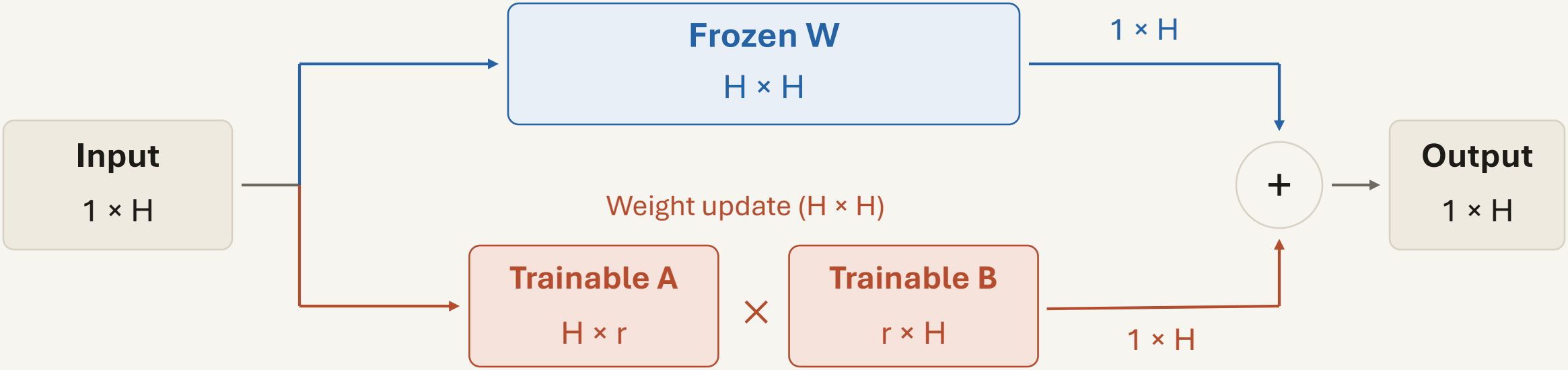
Inference: provide context and choose tokens

Generation repeatedly predicts and appends a token.

Efficient adaptation with LoRA

H: layer width
r: rank, much smaller than H
W: frozen pretrained weights
A, B: trainable matrices

LoRA = Low-Rank Adaptation



Train 2Hr parameters instead of H^2, for this layer.

Adapt the model by learning a small update.

Hu et al. (2022)

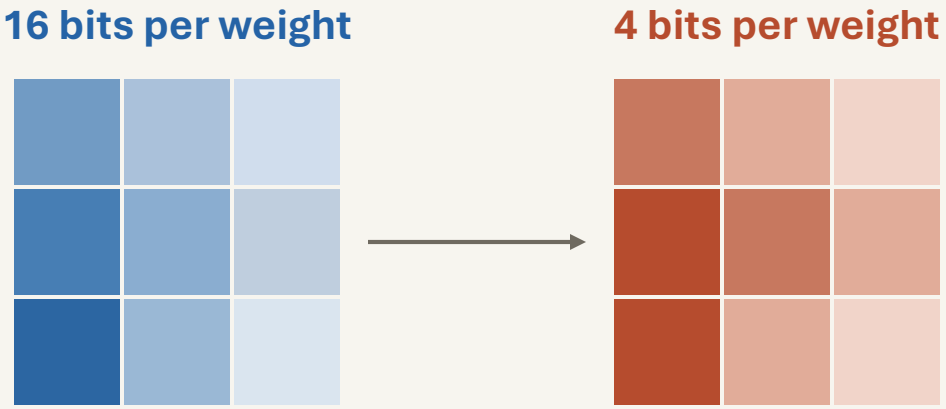
T: sequence length

V: shared vocabulary size

Quantization and distillation

Quantization

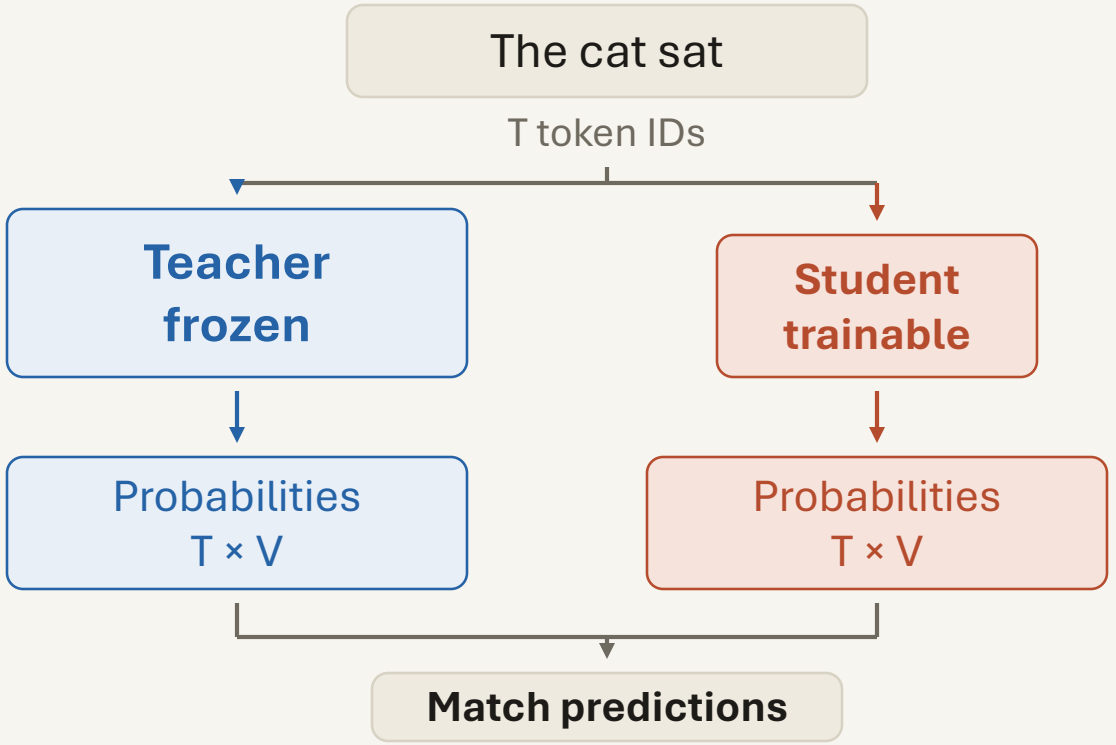
Fewer bits per weight



Same parameter count, lower precision

Frantar et al. (2023)

Distillation

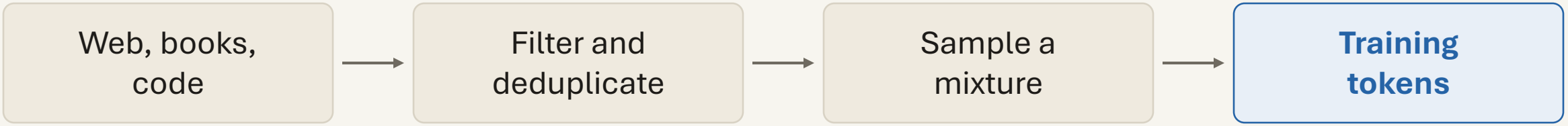


Hinton et al. (2015)

Quantization reduces precision. Distillation transfers behaviour.

N: number of model parameters
D: training tokens processed
C: training compute

Pretraining data and scaling (Scaling Laws for LLMs)



A compute budget

$$C \propto ND$$

Rough approximation for dense Transformers

Larger models cost more per training token.

Balance model size and training data.

Similar training compute

Model	Parameters	Training tokens
Gopher	280 billion	300 billion
Chinchilla	70 billion	1.4 trillion

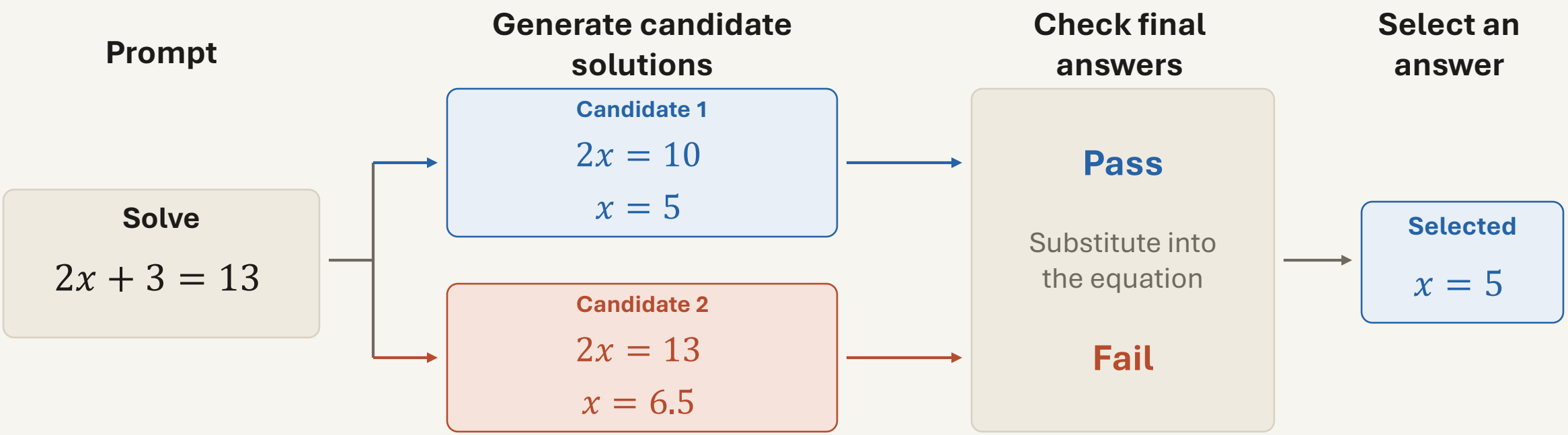
Chinchilla achieved better overall performance.

Kaplan et al. (2020)

Hoffmann et al. (2022)

RLVR: reinforcement learning with verifiable rewards

Reasoning and inference computation (CoT)



Inference: more steps / more candidates; weights fixed

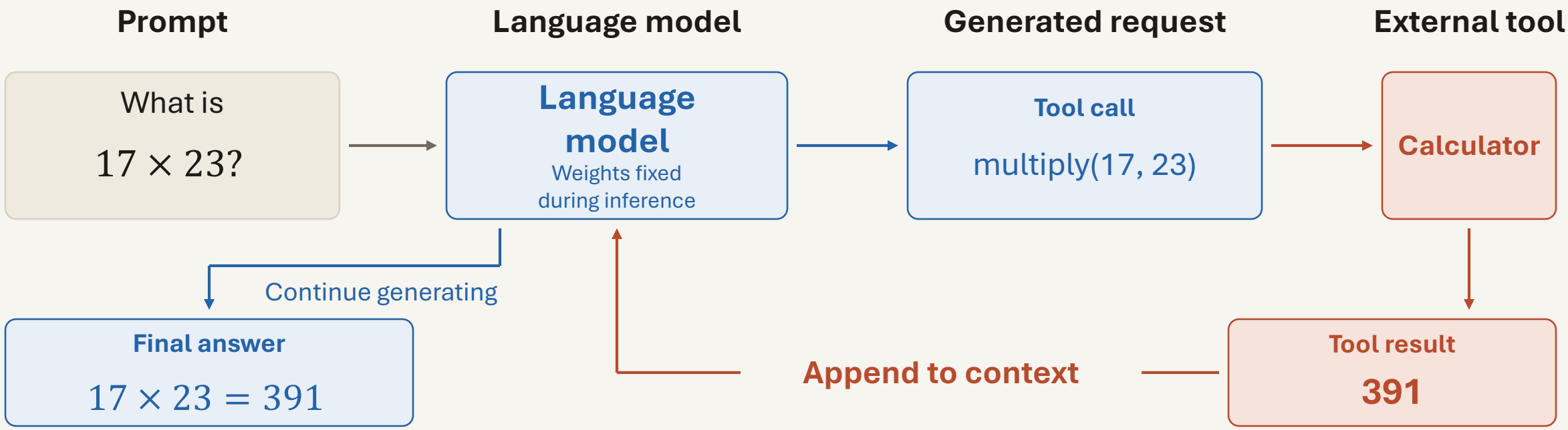
Training (RLVR): checker reward → update weights

Inference compute is another resource.

Wei et al. (2022)

DeepSeek-AI et al. (2025)

Tool use and agents

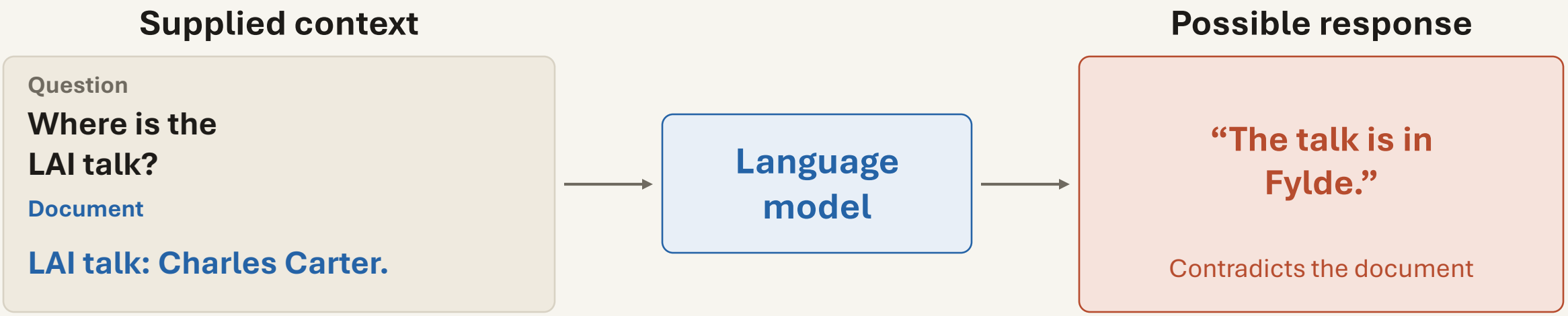


Agent workflow: repeat the loop to work towards a goal.

The model generates requests. External tools execute them.

Toolformer (2023) ReAct (2023)

Hallucinations and uncertainty



Uncertainty

Next-token probabilities predict wording.
Answer confidence needs calibration.

Distribution shift

New domains, languages, or facts can change reliability.

Fluent answers can contradict evidence.

Min et al. (2023)

Lin et al. (2022)