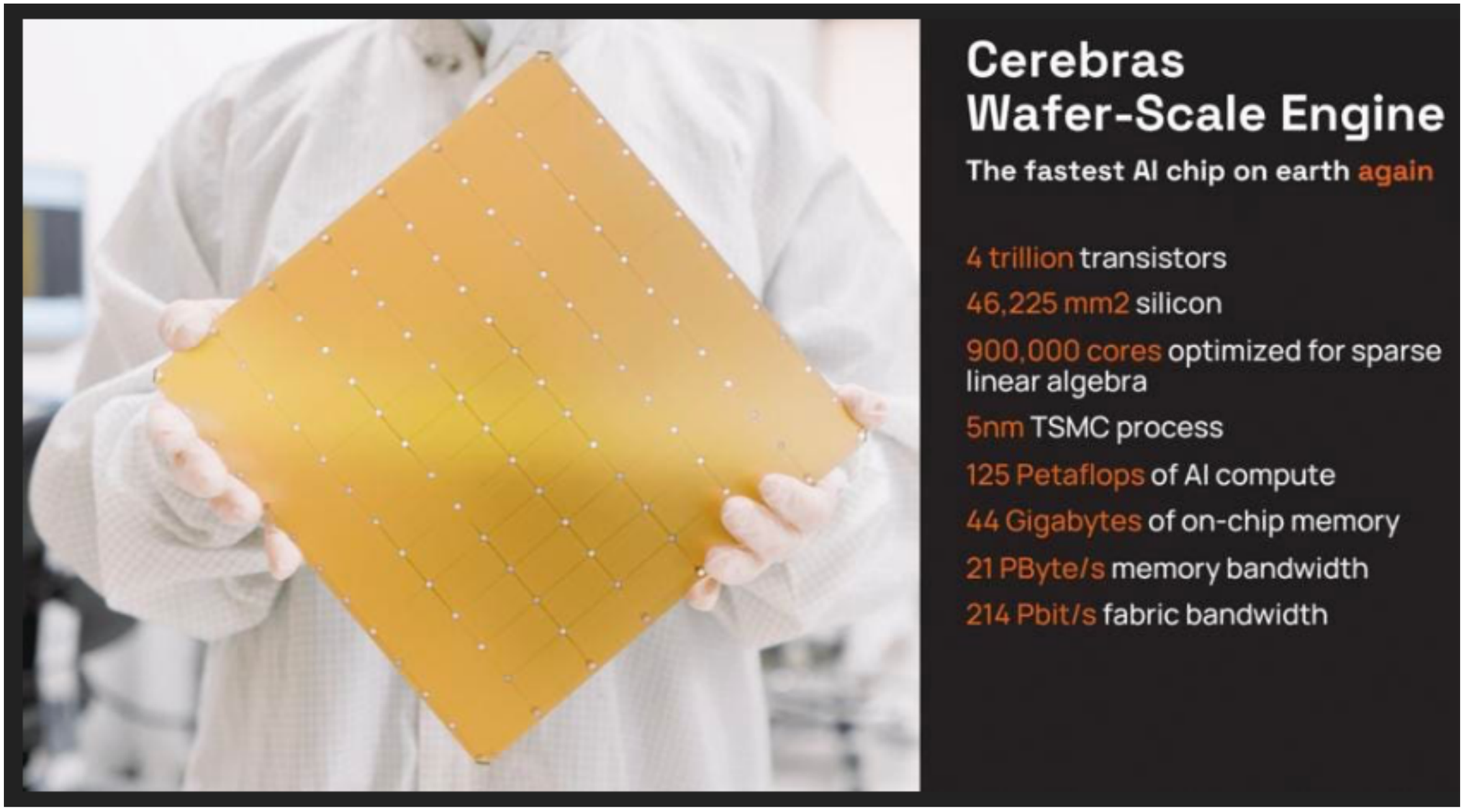
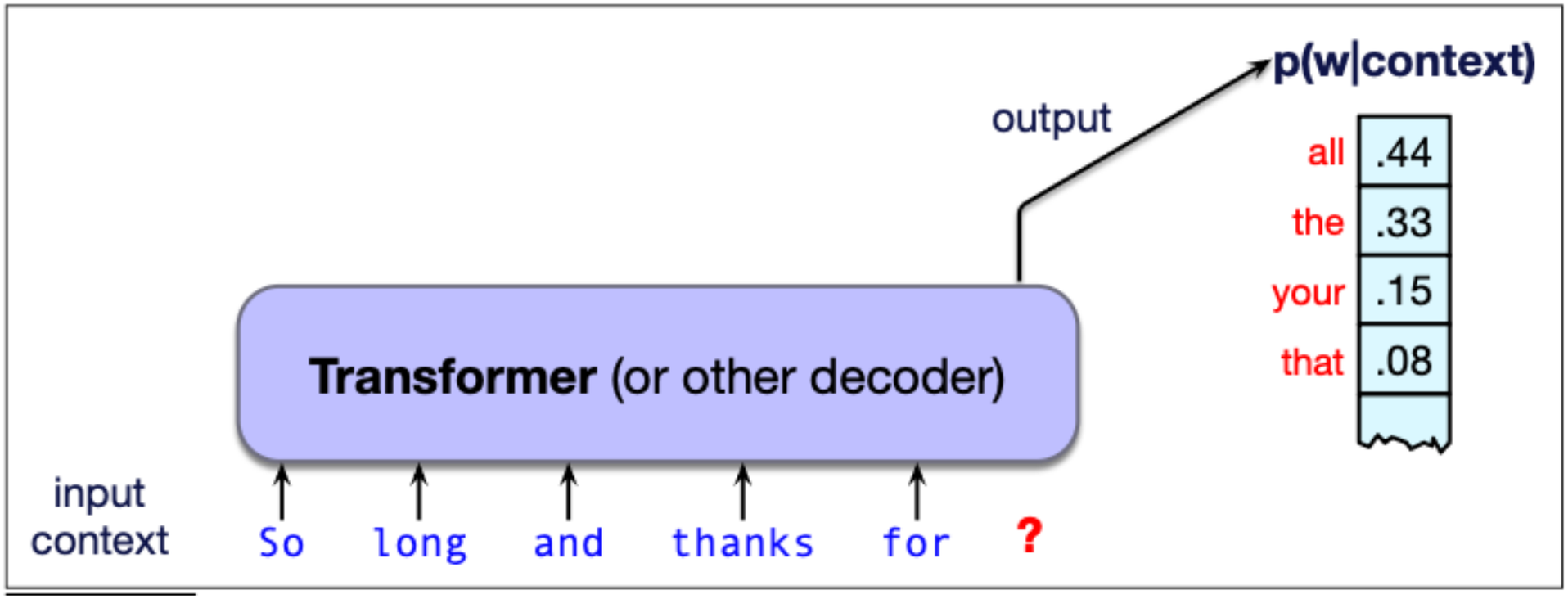
CS 668 Applied Large Language Models
Spring Semester, 2026
Doc 11 Transformer Overview
Feb 17, 2026

Open AI - 6X faster



Language Model

System that can predict next word

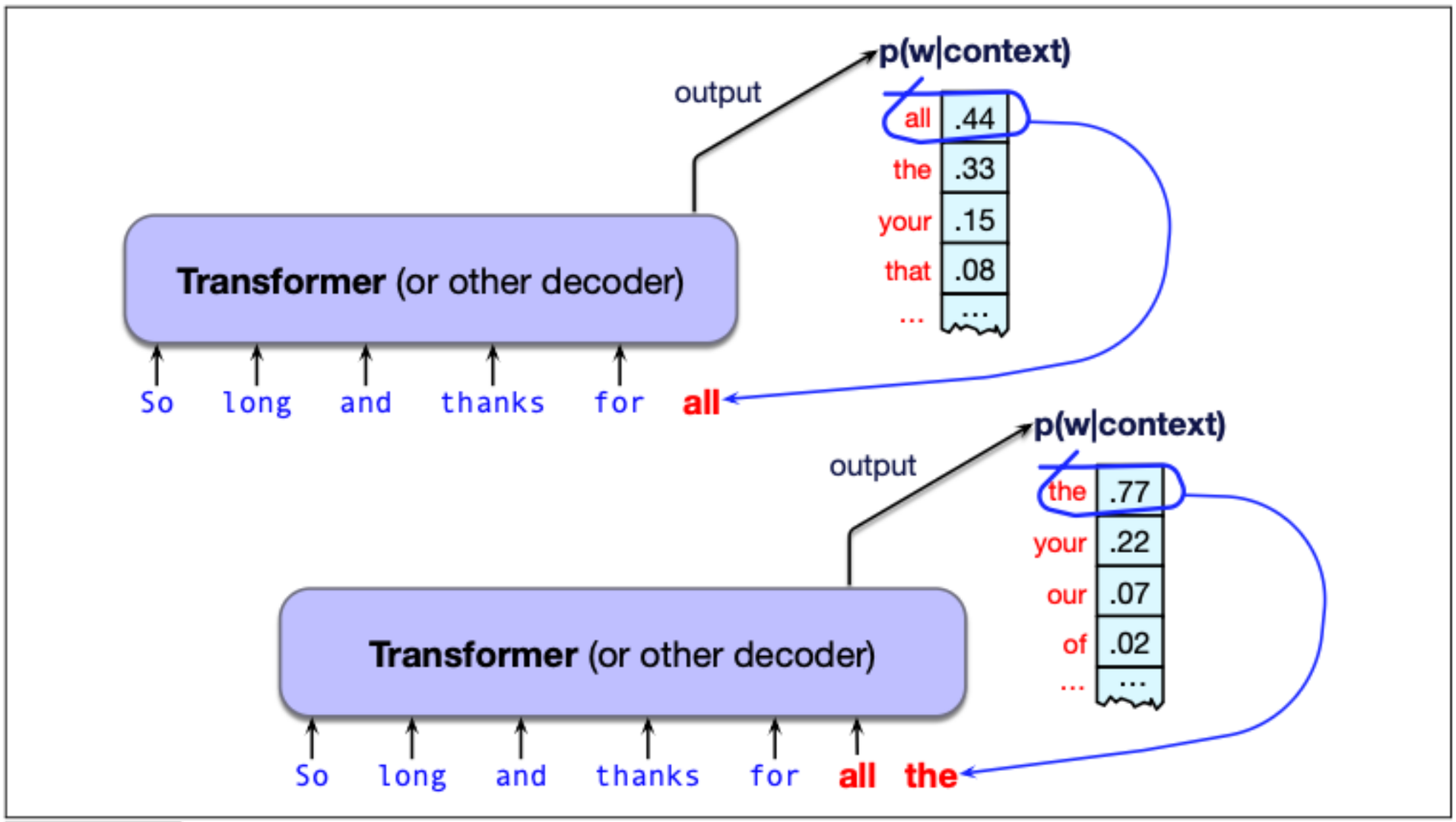


Speech and Language Processing, Jurafsky & Martin, Chapter 7

Autoregressive Language Model

Left-to-right

Causal

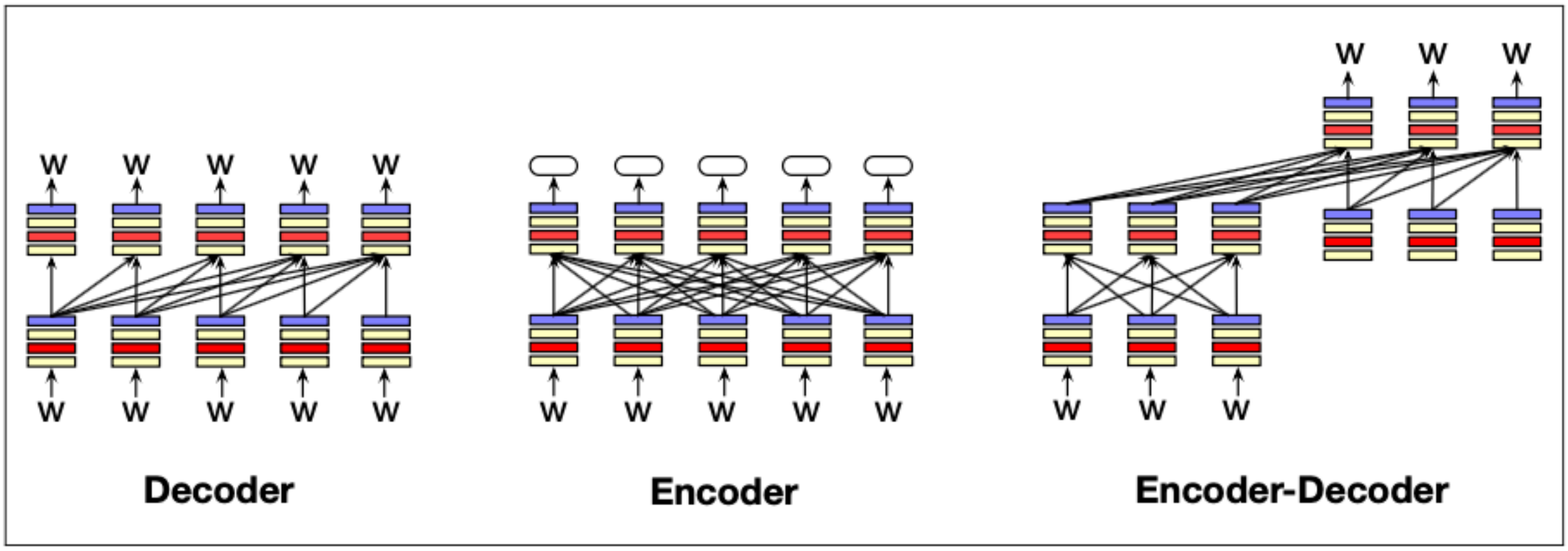


Speech and Language Processing, Jurafsky & Martin, Chapter 7

Three architectures for language models

Decoder

- Iteratively generate output tokens
- ChatGPT
- Claude
- Etc.



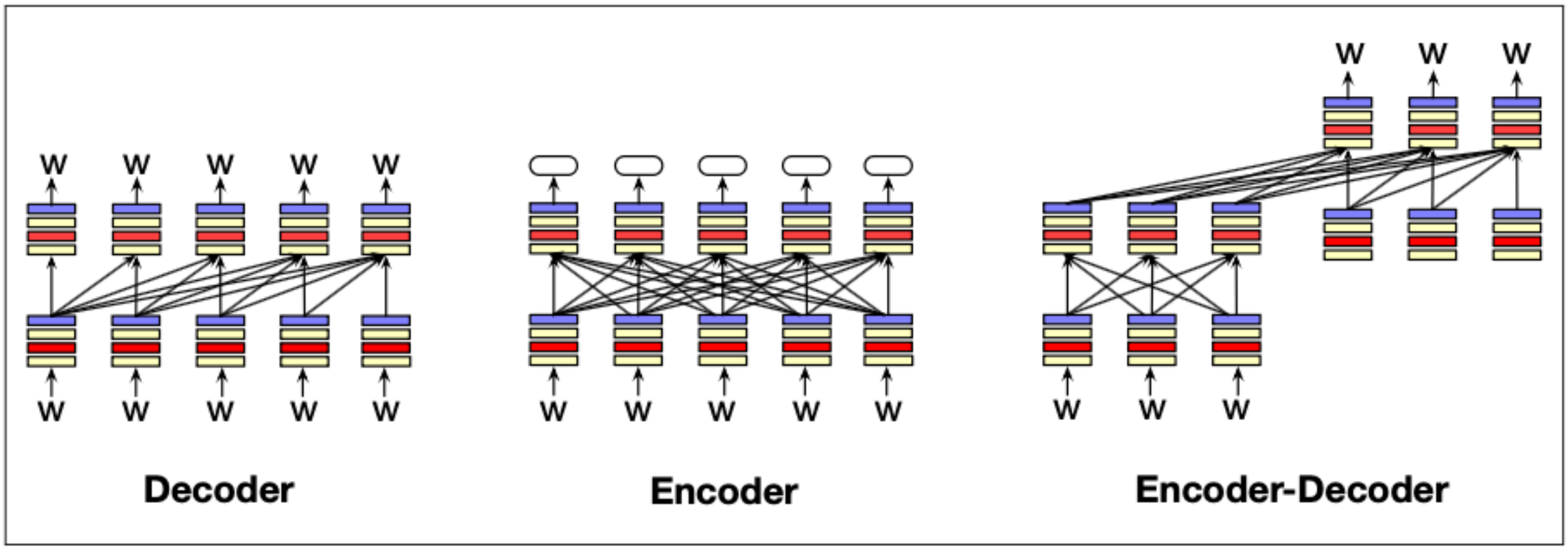
Three architectures for language models

Encoder

- Outputs vector representation for each token
- BERT family
- Used for classifiers

Masked language models

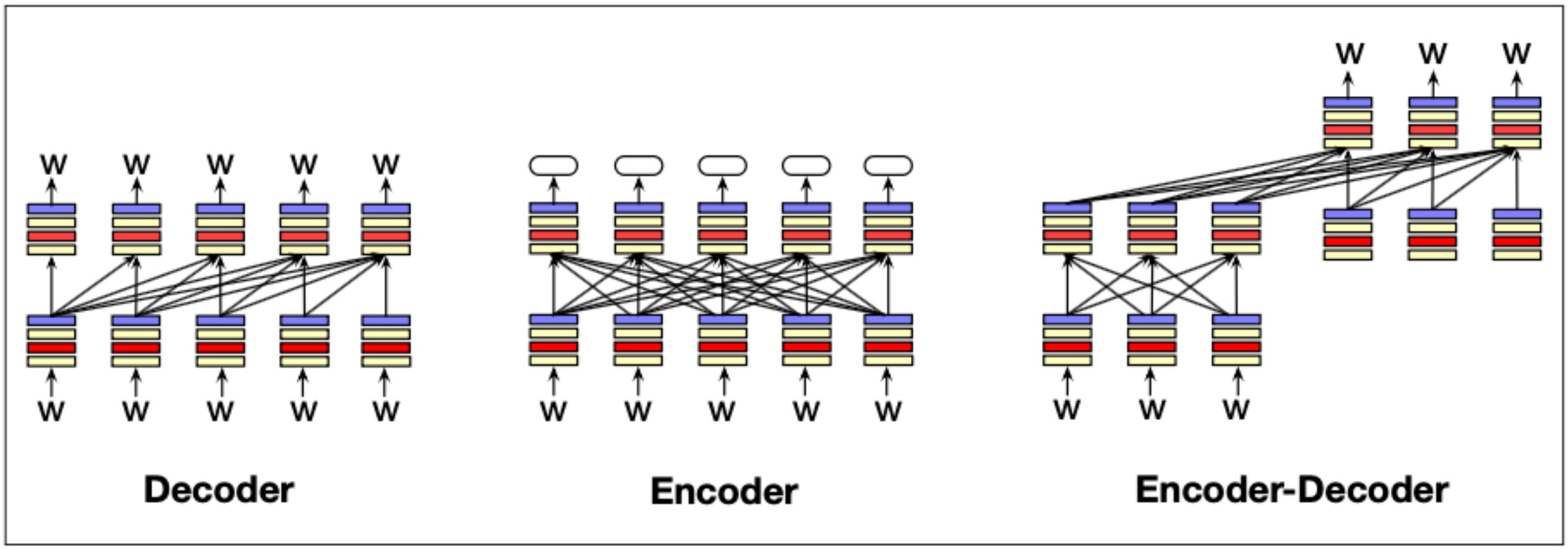
- Trained by masking out a word
- Predict the word



Three architectures for language models

Encoder-decoder

- Input a sequence of tokens and outputs a series of tokens
- Maps between different kind of tokens
- Machine translation
- Speech recognition



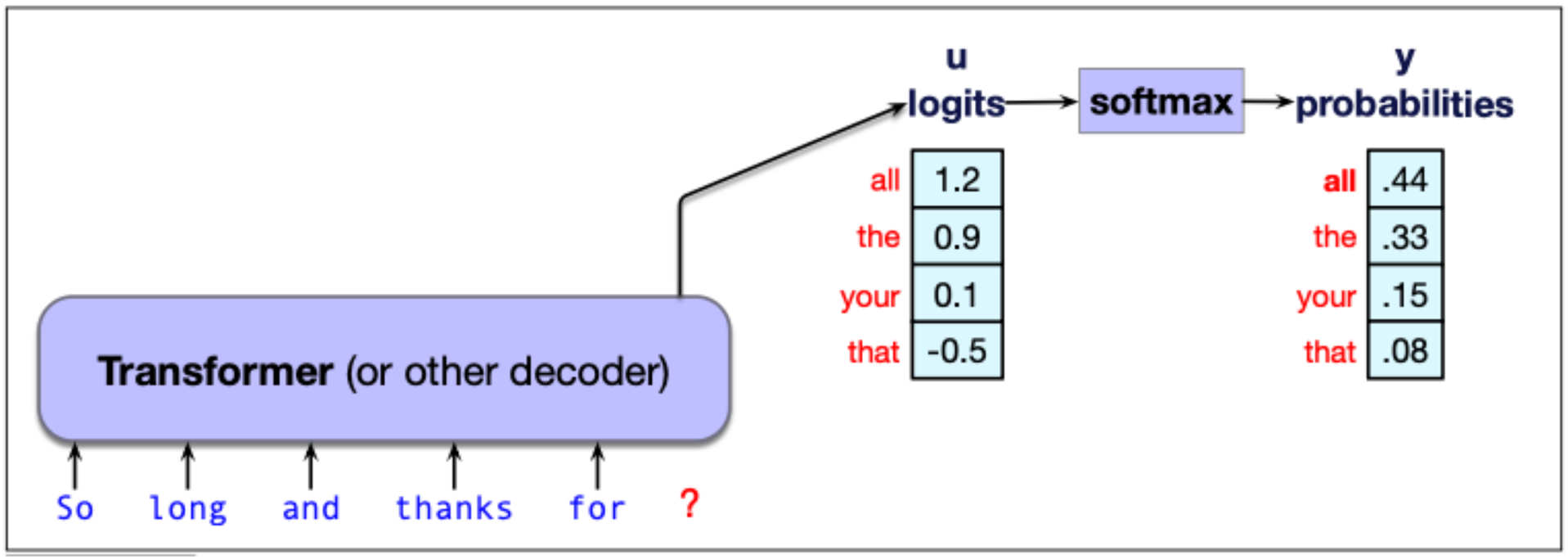
Selecting the Next Token

logits

LLM generates a floating-point score for each token in its vocabulary

Convert the logits to probabilities using sofmax

y= softmax(u)



Speech and Language Processing, Jurafsky & Martin, Chapter 7

Softmax

$$z = [z_1, z_2, ..., z_n]$$

[2.0, 1.0, 0.1]

$$softmax(z_i) = \frac{e^{(z_i)}}{\sum_{j=1}^{n} e^{(z_j)}}$$

$$[e^2,\ e^1,\ e^{0.1}] \approx [\mathbf{7.39},\ \mathbf{2.72},\ \mathbf{1.11}]$$

7.39+2.72+1.11=11.22

[0.66, 0.24, 0.10]

Decoding

Selecting the token based on model's probabilities

Greedy decoding

Just pick the token with the highest probabilty

Prompt

Explain softmax

, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and softmax, and

Gemma-3-12b

Temperature 1

Top K 1

Repeat Penalty 0

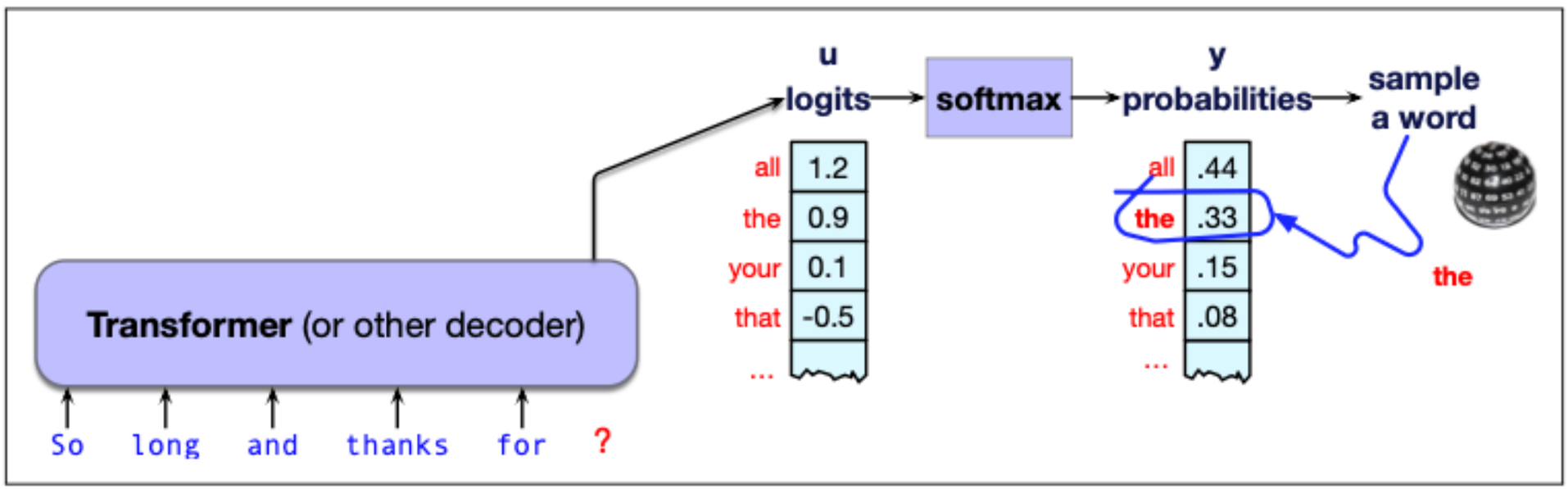
Decoding

Random Sampling

Select tokens randomly based on the probability

There are a lot of low-probability tokens

The likelyhood of one being selected is too high



Vocabulary Size

The number of tokens the model knows

	Typical Size	Examples
English only	30,000 - 60,000	GPT-2
Multilingual	100,000-130,000	GPT-4, Llama 3
	256,000	Gemma

Gemma has tokens for cities, sports teams, emojis

Gemma

Needs to compute probabilities for each token

Embedding Matrix

For each token need the embedding vector

Embedding dimension

- 768 - 4096 common
- 8192+ for large frontier models

Embedding Matrix size

- Number of rows = number of tokens
- Number of columns = embedding dimension

For large models

- 256,000 by 8,192

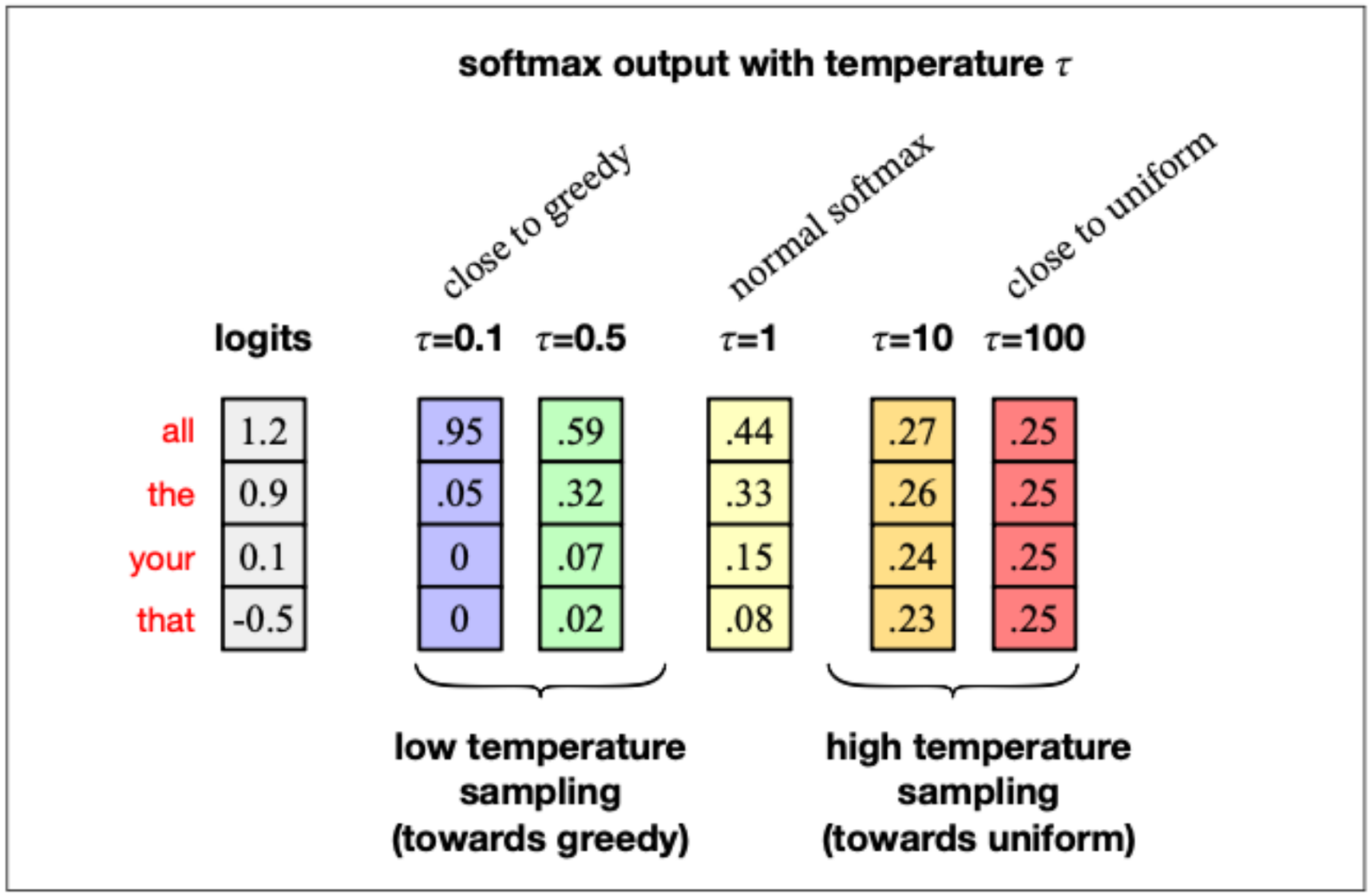
Decoding

Temperature Sampling

Divide the logits by temperature parameter

$$\mathbf{y} = \text{softmax}(\mathbf{u})$$

$$\mathbf{y} = \text{softmax}(\mathbf{u}/\tau)$$



Decoding

Top K Sampling

- Fix K
- Select K tokens with the highest probability
- Renormalize the K values
- Select one randomly using the new probabilities

Issue

- Top K may include too many or too few tokens
  - If there only a few high probability tokens will include some low probability tokens
  - If probabilty distribution is very flat will exclude likely tokens

Decoding

Top P Sampling (Nucleus Sampling)

P = .9

Select a threshold

Sort the tokens by probability, highest to lowest

Calculate the cumulative probability

.44	.44
.33	.77
.15	.92
.08	1.0

From top find first token whose cumulative prob is equal or greater than the threshold

Drop all the tokens below it

The remaining form the nucleus

Do random sampling on the nucleus

Decoding

Min P Sampling

Fix P

Select tokens whose probablity is equat or greater than P

Random sample on the selected tokens

P = .2

.44
.33
.15
.08

Can be used with top P sampling to rid nucleus of low probability tokens

Decoding

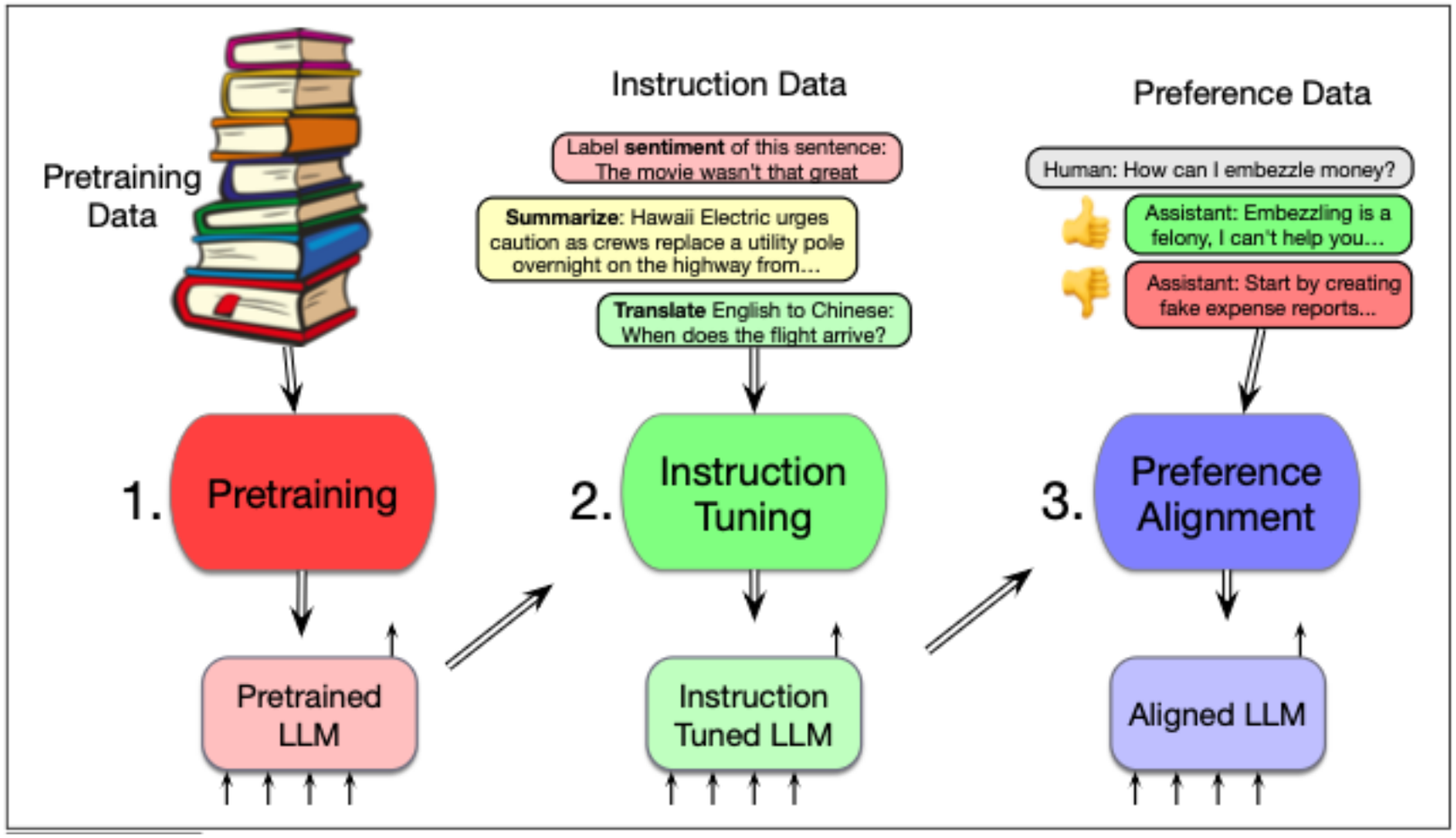
Repeat penalty

Used to reduce a token's probability if it has already appeared in the context window

Usually applied to the logits

$$y_i = \begin{cases} y_i/\theta & \text{if } y_i > 0 \\ y_i \cdot \theta & \text{if } y_i \leq 0 \end{cases}$$

Training LLMs



Speech and Language Processing, Jurafsky & Martin, Chapter 7

Pretraining

- Trained to predict the next token
- Uses cross-entropy loss

Instruction Tuning, Supervised finetuning (SFT)

- Trained to follow instructions
  - Answer questions, write code, give summaries
- Uses cross-entropy loss
- Uses text with instructions and the correct response

Alignment, Preference alignment

- Make it more helpful, less harmful
- Uses reinforcement learning
- Given a context, correct response, bad response

Pretraining - Teacher Forcing, One-Hot Target

Input N tokens, x_1, x_2, … x_N

Have model predict x_{n+1}

Let y = P(x_{N+1}) predicted probability of correct token x_{n+1}

Repeat x_1, x_2, … x_{N+1}

Entropy loss function is -log(y)

For each batch

Compute the average loss

Do backpropagation to adjust

Embeding Matrix is also updated

Gradient $$\frac{\partial L}{\partial z_j} = q_j - y_j$$

z_j be the j-th logit

q_j is the predicted probability

y_j is the ground-truth value

Some Batch Sizes

Phase	Global Batch Size	Purpose
Early Training	~1 Million tokens	Improves stability and early convergence.
Middle Training	~4 Million tokens	Increases throughput and utilizes the 16,000 H100 GPUs efficiently.
Late Training	Up to 16 Million tokens	Used during specific annealing phases or context-length extensions.

Batch & Sequences

Batches are divided into sequences

Llama 3 sequences are 8,129 tokens

Batches are too large to fit into memory

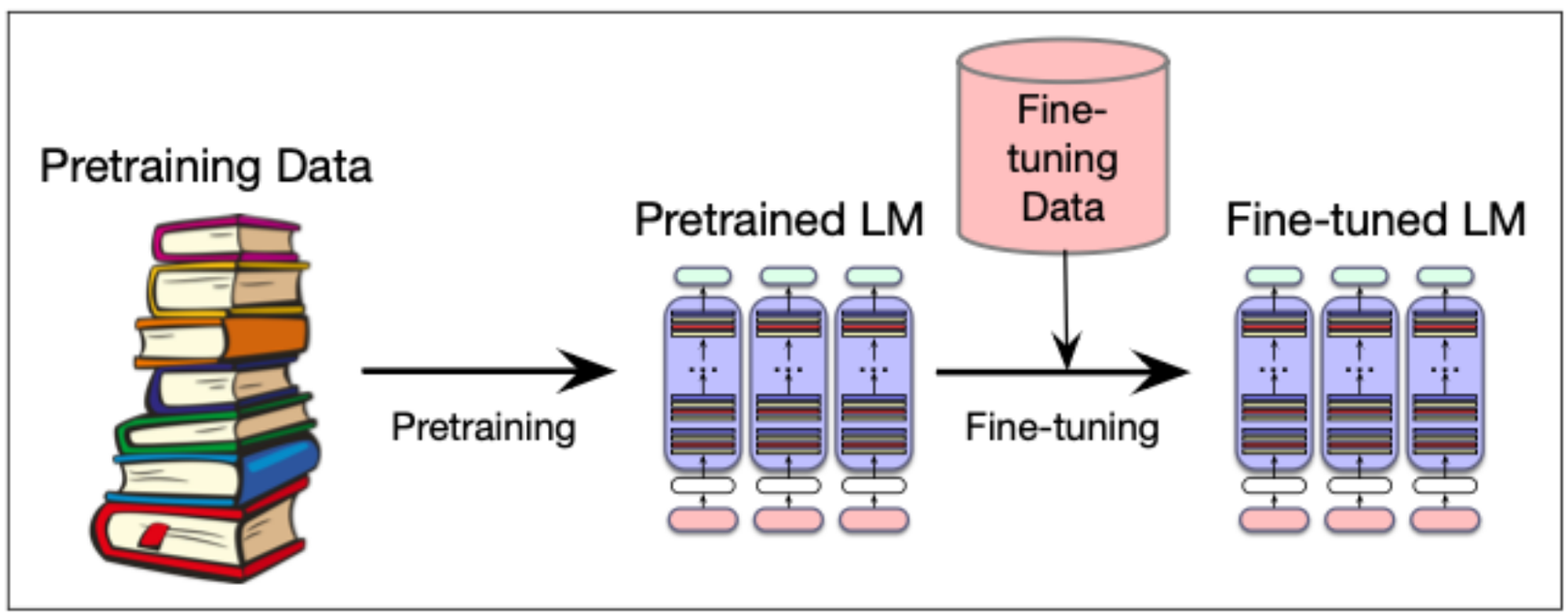
Computing attention is nearly quadratic

Sequences

- Saturate a single GPU
- Sequences can be sent to different GPUs & processed in parallel

Fine-tuning (Continued Pretraining)

Training the LLM on more data, often from a particular domain



Full fine-tuning

- Update all weights

Partial fine-tuning

- Update some weights

Additive (LoRA)

- Add weights

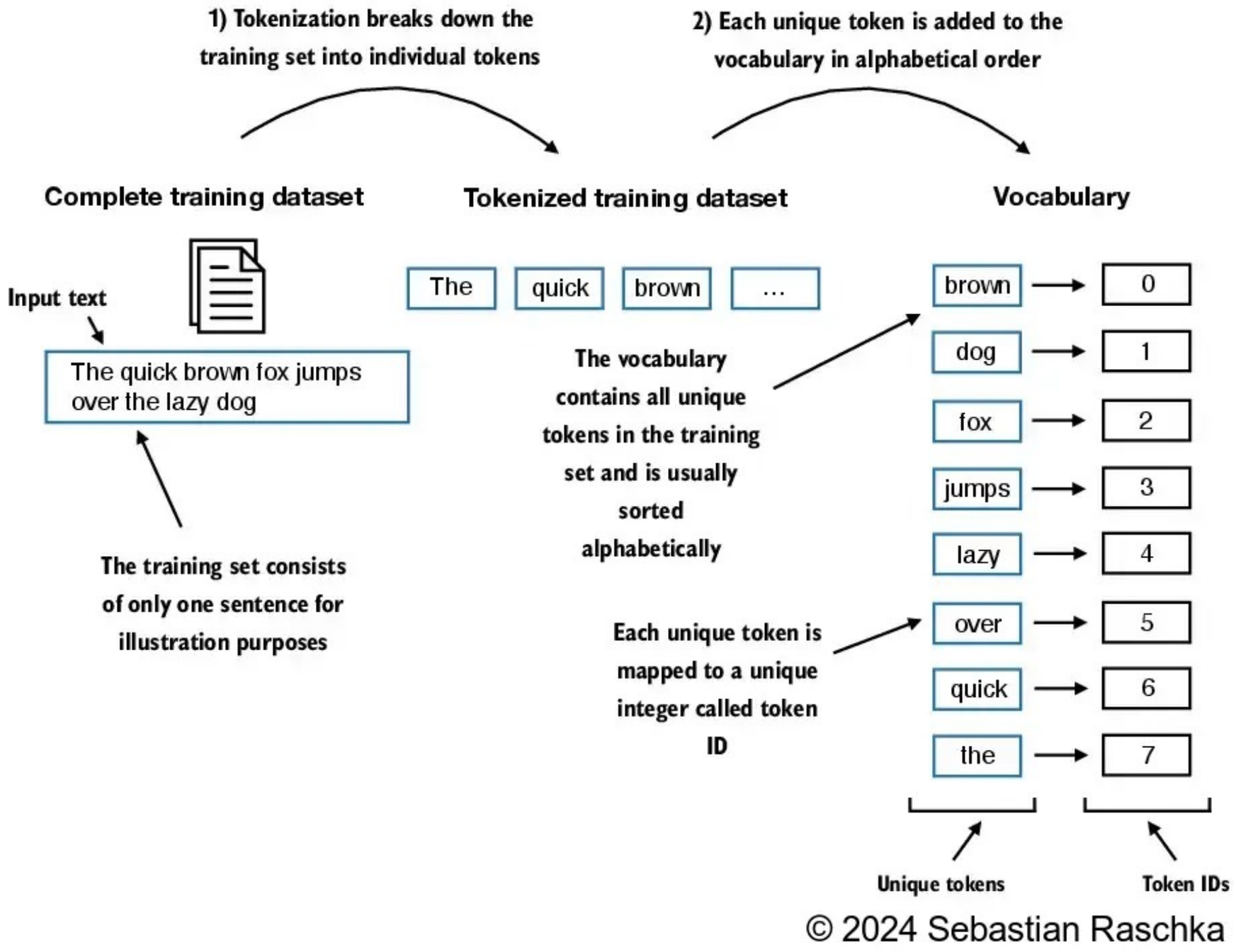
Input

LLMs cannot process raw data directly

- Text
- Sound
- Images
- Video

Text

- Convert text into numbers 0 through n


1) Tokenization breaks down the training set into individual tokens
2) Each unique token is added to the vocabulary in alphabetical order
Complete training dataset
Tokenized training dataset
Vocabulary
Input text
The quick brown fox jumps over the lazy dog
The training set consists of only one sentence for illustration purposes
The
quick
brown
...
The vocabulary contains all unique tokens in the training set and is usually sorted alphabetically
Each unique token is mapped to a unique integer called token ID
brown → 0
dog → 1
fox → 2
jumps → 3
lazy → 4
over → 5
quick → 6
the → 7
Unique tokens
Token IDs
© 2024 Sebastian Raschka

Overview

Create a token dictionary mapping tokens to integers (token ids)

- Collect all words in your possible input
- Tokenize them
- Add special tokens
- Sort the tokens & assign them integers 0 - n

LLM Input

- Tokenize the input
- Use the token dictionary to convert tokens to integers (token ids)
- Sequence of integers is the input to the model
- Output is sequence of token ids
- Use a token dictionary to convert token IDs to text

Each Model has its own Tokenizer

Example string: "antidisestablishmentarianism"

r50k_base: 5 tokens (gpt2)
token integers: [415, 29207, 44390, 3699, 1042]
token bytes: [b'ant', b'idis', b'establishment', b'arian', b'ism']

cl100k_base: 6 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [519, 85342, 34500, 479, 8997, 2191]
token bytes: [b'ant', b'idis', b'establish', b'ment', b'arian', b'ism']

o200k_base: 6 tokens (gpt-40, gpt-40-mini)
token integers: [493, 129901, 376, 160388, 21203, 2367]
token bytes: [b'ant', b'idis', b'est', b'ablishment', b'arian', b'ism']

Example string: "Goodbye, and thanks for all the fish!"

r50k_base: 10 tokens (gpt2)
token integers: [10248, 16390, 11, 290, 5176, 329, 477, 262, 5916, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

cl100k_base: 10 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [15571, 29474, 11, 323, 9523, 369, 682, 279, 7795, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

o200k_base: 10 tokens (gpt-40, gpt-40-mini)
token integers: [17212, 43571, 11, 326, 11707, 395, 722, 290, 13897, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

Root Words

Example string: "fun funny funnier funniest funnel fundamentalist functional"

cl100k_base: 10 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [12158, 15526, 2523, 90083, 2523, 85880, 61319, 16188, 380, 16003]
token bytes: [b'fun', b' funny', b' fun', b'nier', b' fun', b'niest', b' funnel', b' fundamental', b'ist', b' functional']

o200k_base: 9 tokens (gpt-40, gpt-40-mini)
token integers: [18142, 21217, 2827, 25238, 174952, 50625, 18864, 421, 20483]
token bytes: [b'fun', b' funny', b' fun', b'nier', b' funniest', b' funnel', b' fundamental', b'ist', b' functional']

Non-English

Example string: "नमस्ते, आप कैसे हैं"

r50k_base: 31 tokens (gpt2)
token integers: [11976, 101, 11976, 106, 11976, 116, 24231, 235, 11976, 97, 24231, 229, 11, 28225, 228, 11976, 103, 28225, 243, 24231, 230, 11976, 116, 24231, 229, 28225, 117, 24231, 230, 11976, 224]

o200k_base: 8 tokens (gpt-40, gpt-40-mini)
token integers: [998, 1637, 14681, 628, 11, 9717, 63073, 6374]

Token Limits & Cost - Open AI

Model	Input Context Window - Tokens	Output limit - Tokens	Speed Tokens per second	Cost per Million Tokens
GPT-3.5 Turbo	16,385	4,096	121.5	Input $0.50 Output $1.50
GPT-4 Turbo	128,000	4,096	39.3	Input $10 Output $30
GPT-4o	128,000	16,384	134.9	Input $2.50 Output $10
o1-preview	128,000	32,000	151.3	Input $15 Output $60

The Verdict.txt - Chapter two

20479 Characters
3,747 Words

r50k_base: 5145 tokens

p50k_base: 5145 tokens

cl100k_base: 4943 tokens

o200k_base: 4836 tokens

tiktoken https://github.com/openai/tiktoken

Fast BPE tokeniser for use with OpenAI's models

pip install tiktoken

```
import tiktoken
enc = tiktoken.get_encoding("o200k_base")
assert enc.decode(enc.encode("hello world")) == "hello world"

# To get the tokeniser corresponding to a specific model in the OpenAI API:
enc = tiktoken.encoding_for_model("gpt-4o")
```

Source Code for Token Examples

```
import tiktoken

def compare_encodings(example_string: str) -> None:
    """Prints a comparison of three string encodings."""
    # print the example string
    print(f'\nExample string: “{example_string}"')

    # for each encoding, print the # of tokens, the token integers, and the token bytes
    for encoding_name in ["r50k_base", "cl100k_base", "o200k_base"]:
        encoding = tiktoken.get_encoding(encoding_name)
        token_integers = encoding.encode(example_string)
        num_tokens = len(token_integers)
        token_bytes = [encoding.decode_single_token_bytes(token) for token in token_integers]
        print()
        print(f"{encoding_name}: {num_tokens} tokens")
        print(f"token integers: {token_integers}")
        print(f"token bytes: {token_bytes}”)
```

Downloading a Model

```
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(
    "microsoft/Phi-3-mini-4k-instruct",
    attn_implementation='eager',
    torch_dtype="auto",
    trust_remote_code=True,
)
tokenizer = AutoTokenizer.from_pretrained("microsoft/Phi-3-mini-4k-instruct")
```

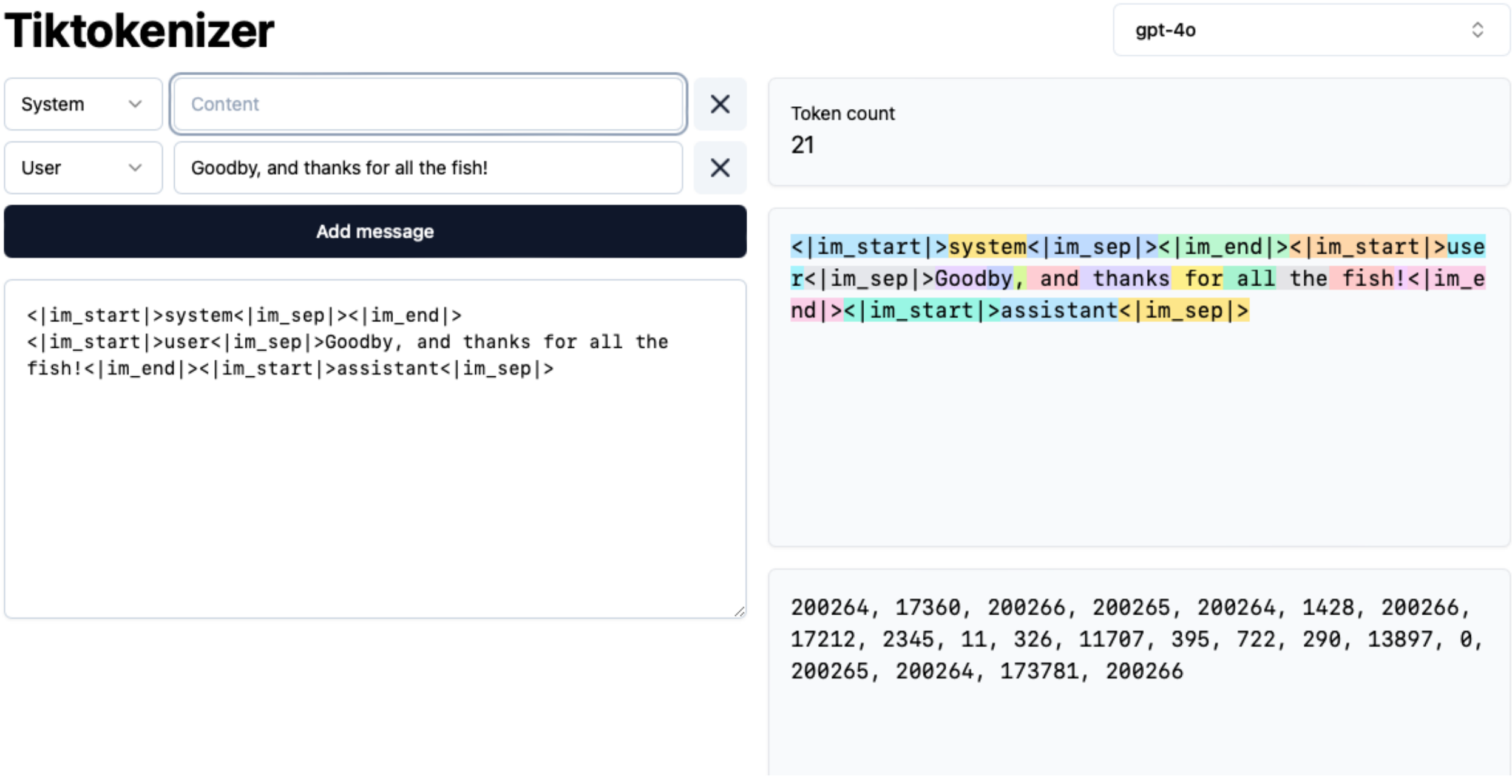
A Pipeline

```
from transformers import pipeline


# Create a pipeline
generator = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    return_full_text=True,
    max_new_tokens=500,
    do_sample=False
)
```

https://tiktokenizer.vercel.app/

Online tokenizer



Tokenizer Issues

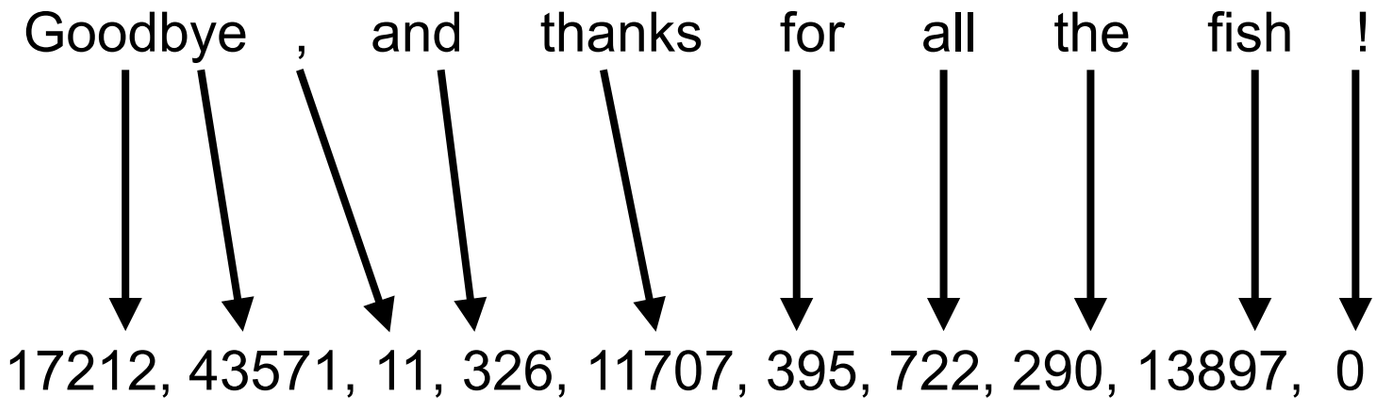
What is a token

Special Characters

Missing words in the token dictionary

<|unk|>

The Mapping



Tokens are just a way to represent works in numbers

They don’t capture the relationship between words

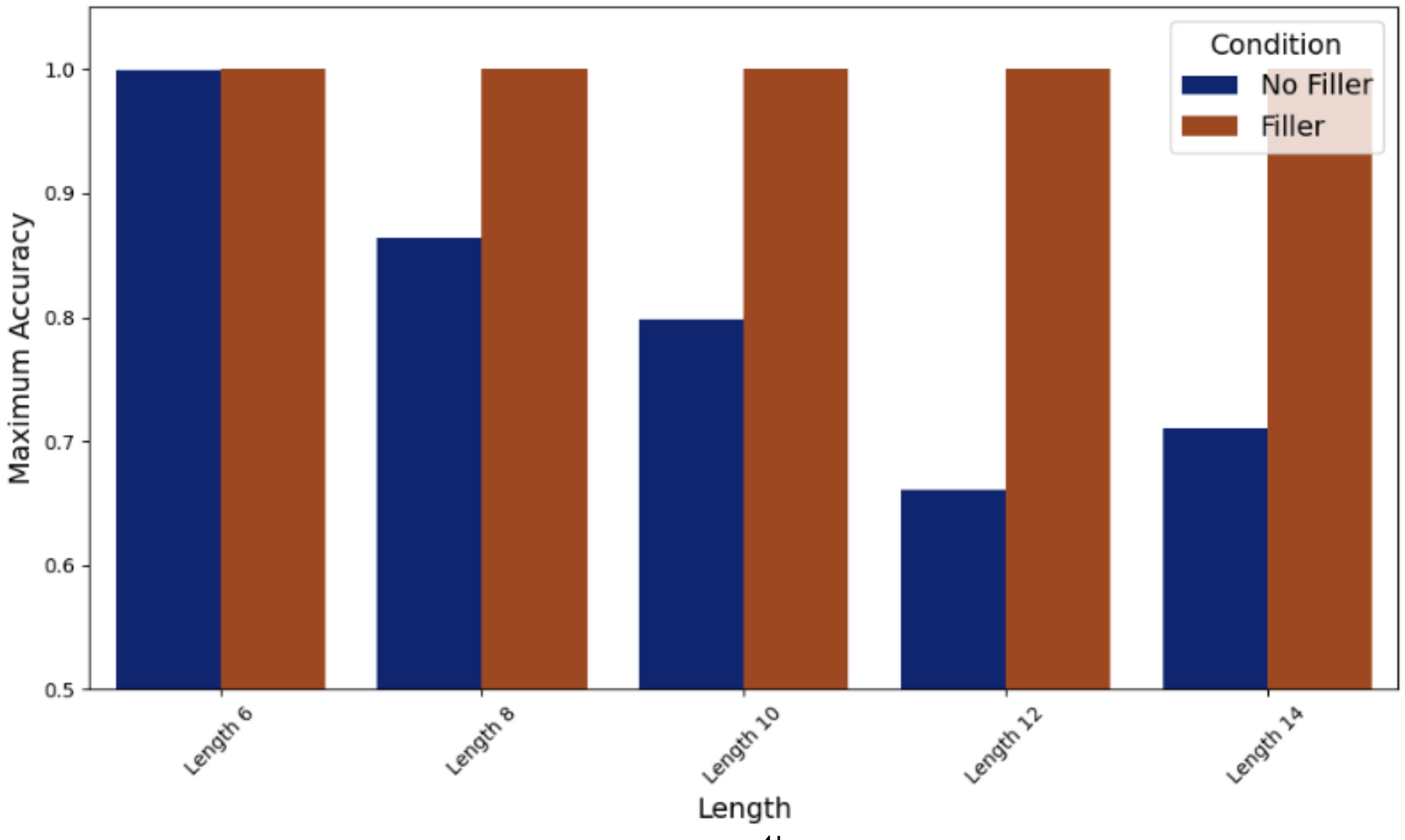
To capture the relationship between tokens, convert them to a vector

Filler Tokens

Let's Think Dot by Dot:

Hidden Computation in Transformer Language Models

Transformers can use meaningless filler tokens (e.g., '.....') in place of a chain of thought to solve two hard algorithmic tasks



Byte Pair Encoding (BPE)

Original Algorithm (for compression)

- Find the most common pair of characters
- Replace with new symbol
- Repeat

Example from Wikipedia

aaabdaaabac

ZabdZabac
Z=aa

ZYdZYac
Y=ab
Z=aa

XdXac
X=ZY
Y=ab
Z=aa

Byte-level BPE

Used by BERT models, GPT-2

When coming across words not in the vocabulary

Convert to UTF-8 and encode pairs of characters

No need for <|unk|>

```
import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")
text = "aaabdaaabac"
integers = tokenizer.encode(text)
for i in integers:
    print(f"{i} -> {tokenizer.decode([i])}")
```

"aaabdaaabac"

7252 -> aa
397 -> ab
6814 -> da
64 -> a
397 -> ab
330 -> ac

“This is a cat”

1212 -> This
318 -> is
257 -> a
3797 -> cat

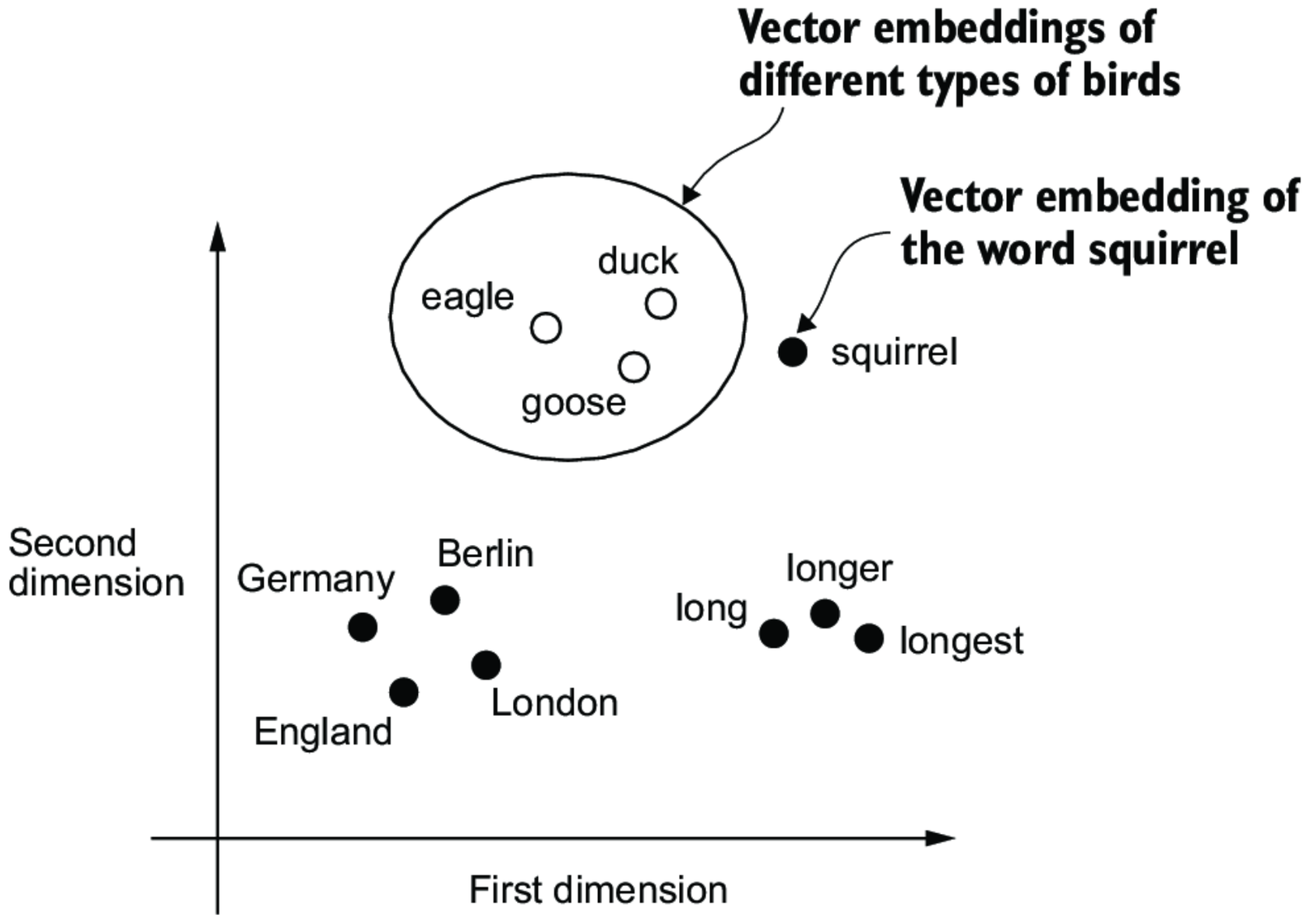
"Thisisacat"

1212 -> This
271 -> is
330 -> ac
265 -> at

Token to Embeddings

Represent a token as a vector in n-space

Related tokens should be close to each other



Hands on Large Language Models, Jay Alammar and Maarten Grootendorst

Type of Embeddings

Token Embeddings

Vector representation (embedding) using a lookup table

Segment Embeddings

Which sentence a token belongs to

Position Embeddings

Position of each token in the sequence

Embedding Space

Higher the dimension of the space

- Captures more information about the relationship between tokens
- Requires more computation

Embedding Size

GPT-2 Models	768 dimensions
GPT-3 (175 parameters)	12,288 dimensions
Bert-base	768 dimensions
Bert-large	1024 dimensions

Map each token to a vector in the space

How to do the Embedding

N number of tokens

D dimension of embedding space (hyperparameter)

Create a matrix (weight matrix) with N rows and D columns

Fill with random values

The K’th row is the embedding (vector) of token ID K

Use training data to modify the weight matrix

Example

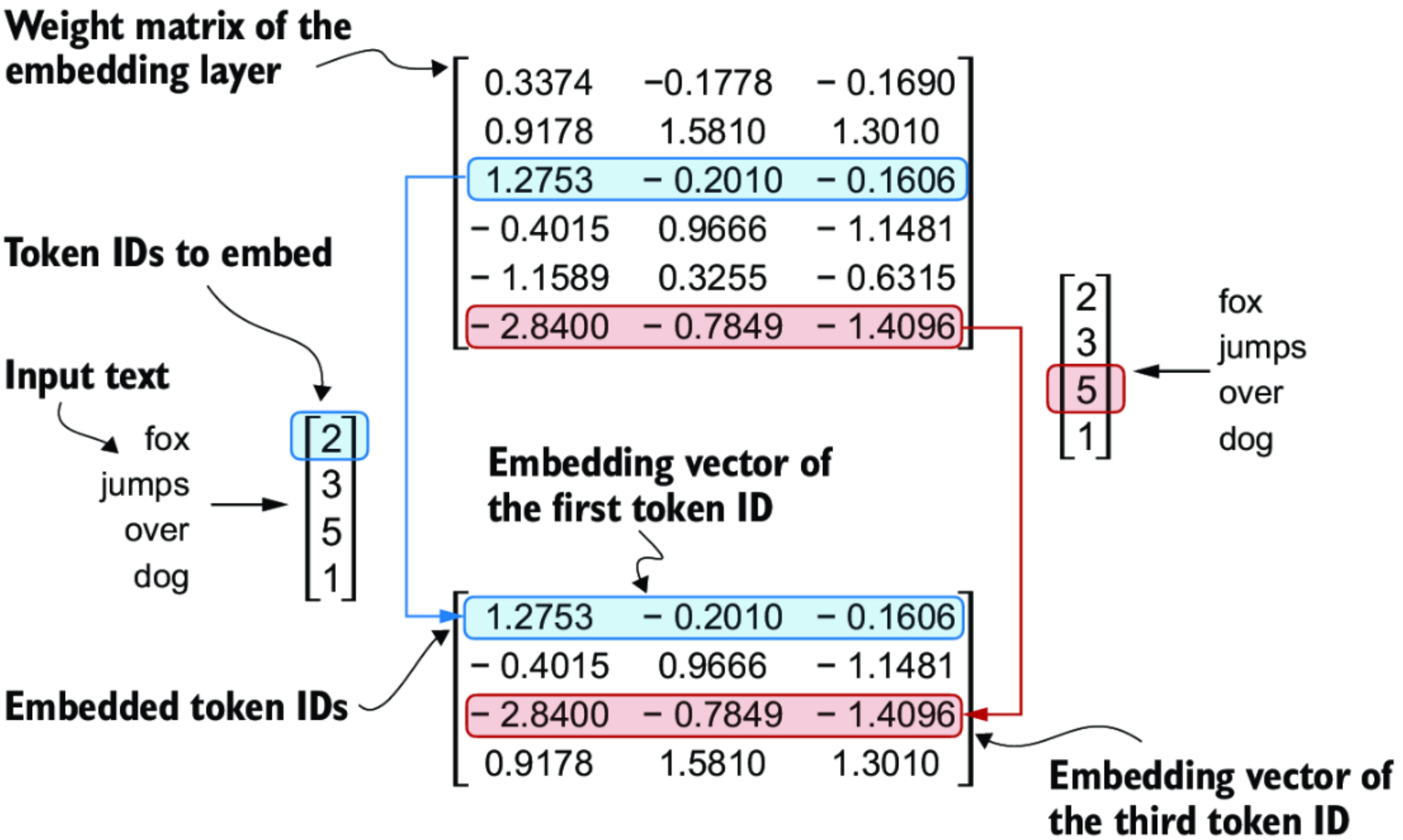
Small values so can see what is going on

Number of tokens = 6

Dimension of embedding space = 3

Input text

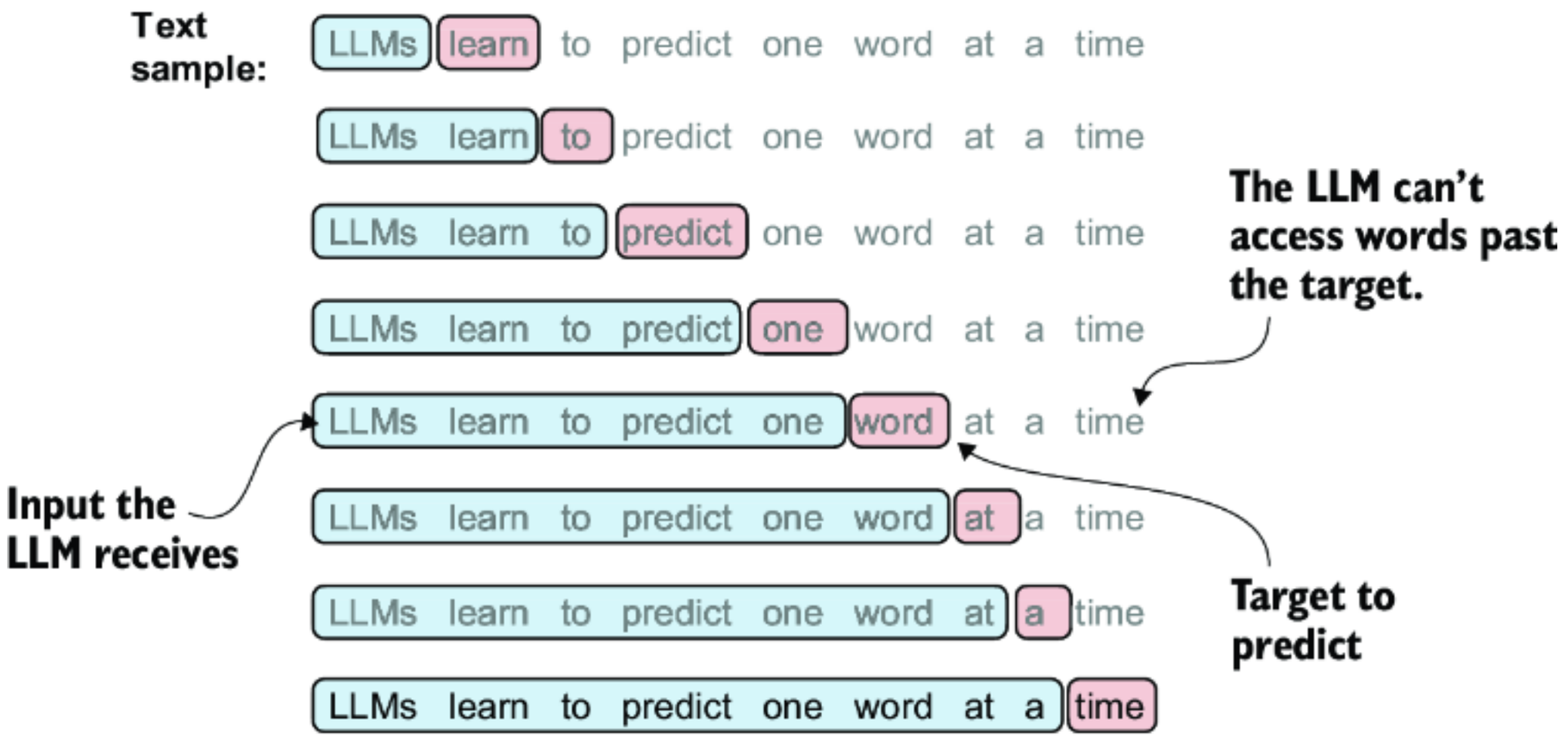
Fox jumps over dog



Hands on Large Language Models, Jay Alammar and Maarten Grootendorst

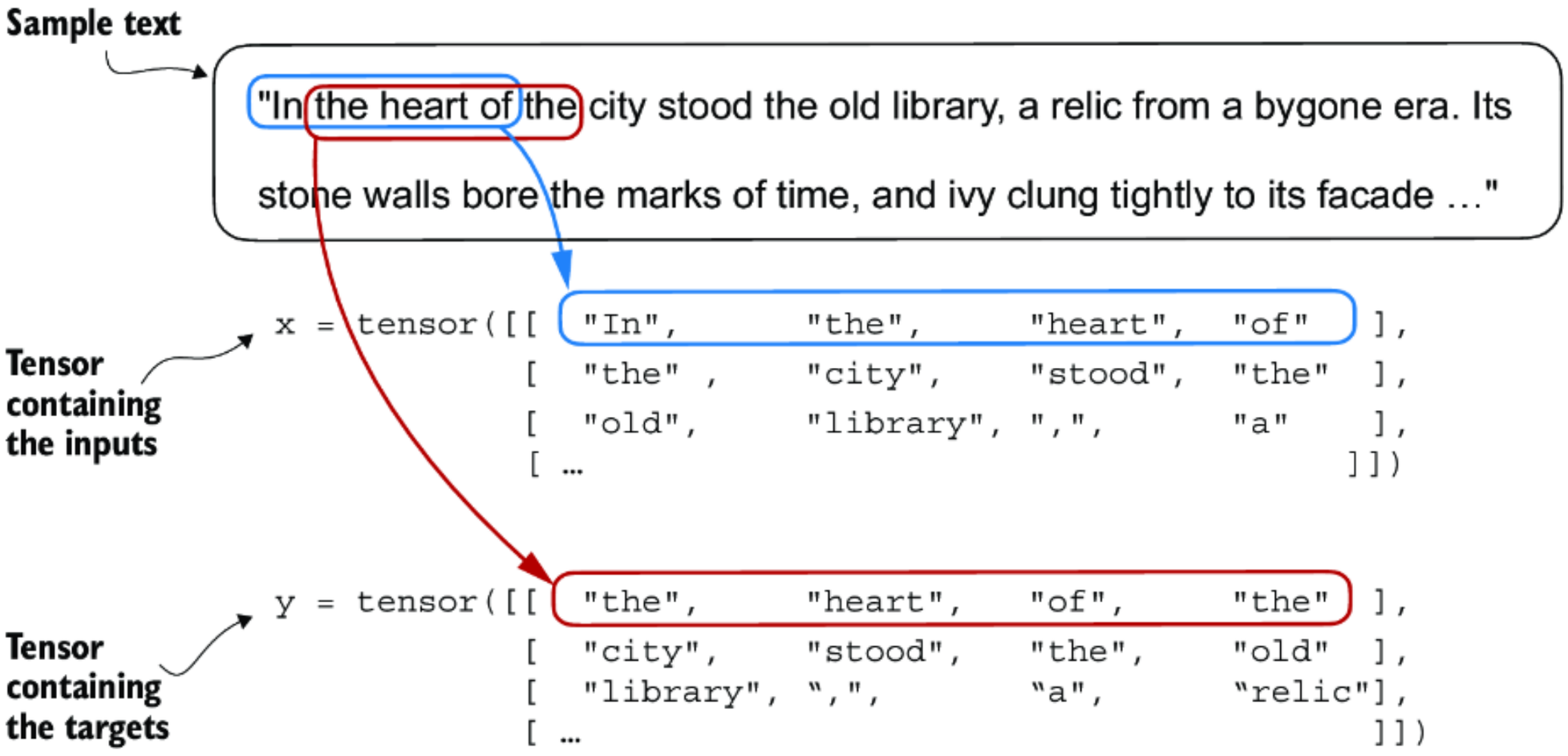
LLM Predict the next Word

Training



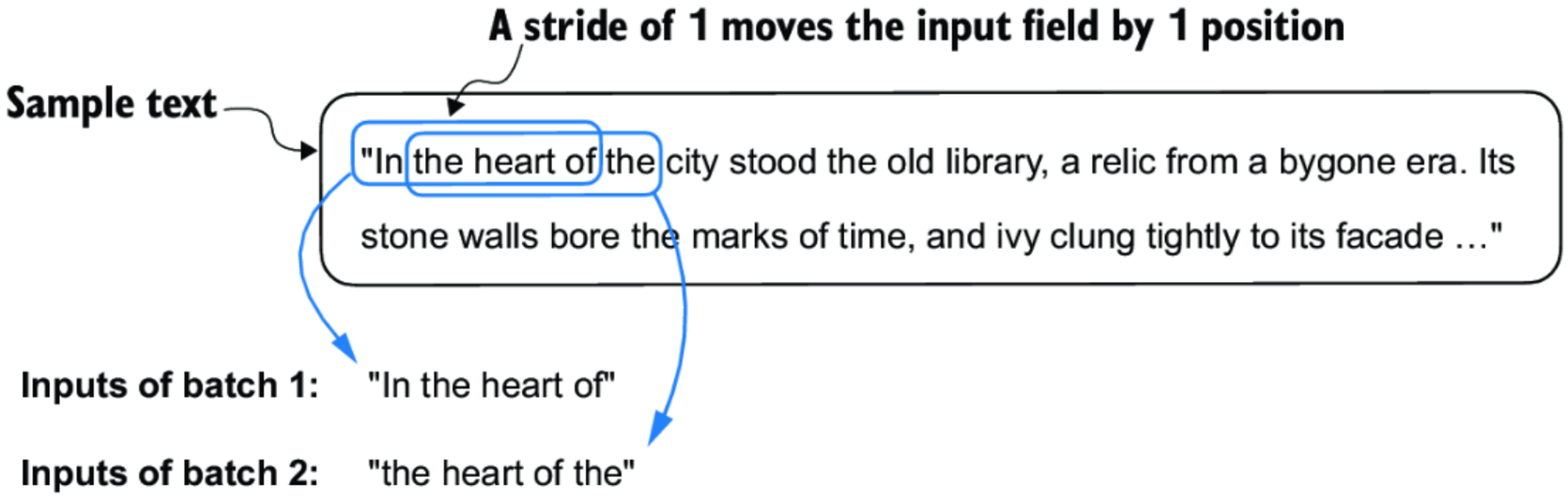
Hands on Large Language Models, Jay Alammar and Maarten Grootendorst

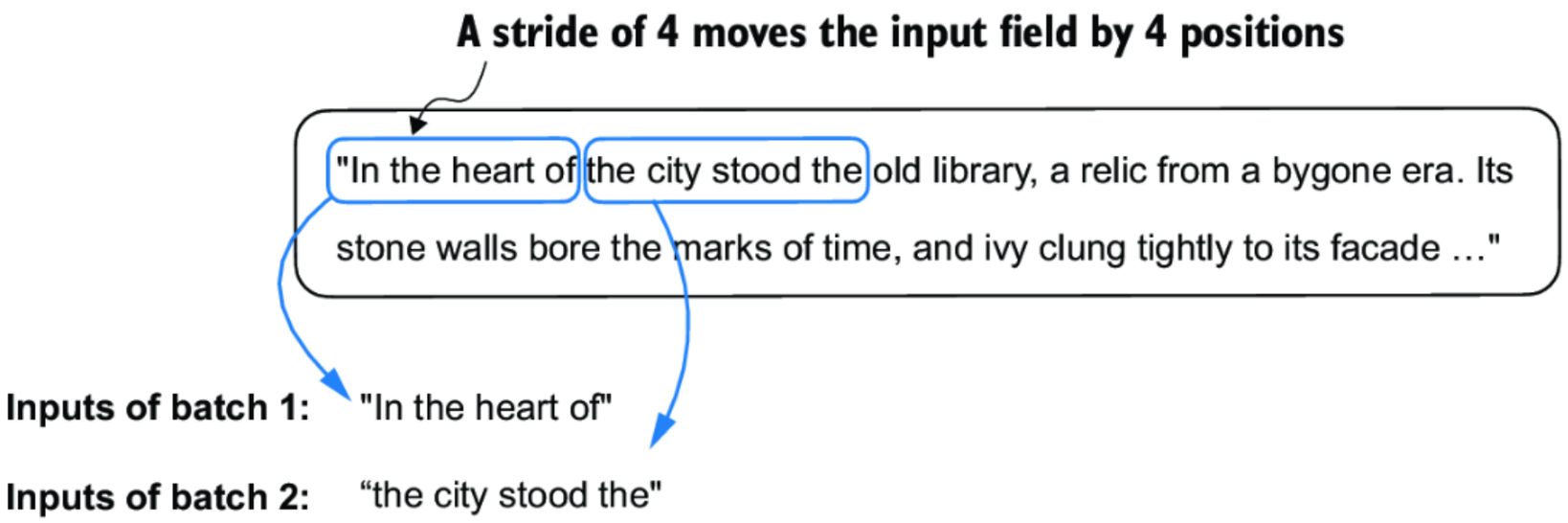
Training



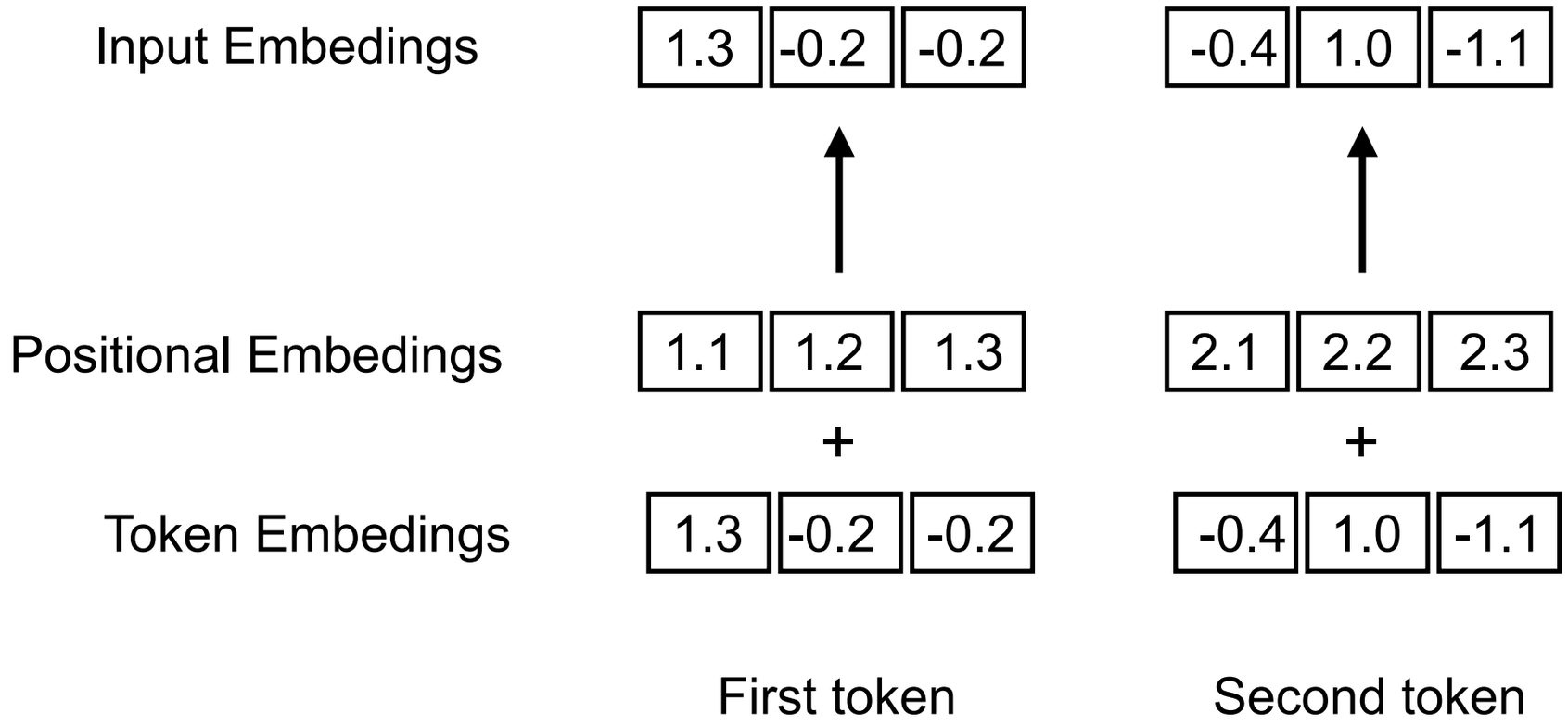
Hands on Large Language Models, Jay Alammar and Maarten Grootendorst

Stride, Window Size, Context





Adding Absolute Positional Embeddings



GPT uses absolute positional embeddings optimized in training

Positional Embeddings - Sinusoidal

Consider the sentence: "The cat sat on the mat."

Position 1 ("The"): [sin(1/10000^(0/5)), cos(1/10000^(0/5)), sin(1/10000^(2/5)), cos(1/10000^(2/5))]
Position 2 ("cat"): [sin(2/10000^(0/5)), cos(2/10000^(0/5)), sin(2/10000^(2/5)), cos(2/10000^(2/5))]
Position 3 ("sat"): [sin(3/10000^(0/5)), cos(3/10000^(0/5)), sin(3/10000^(2/5)), cos(3/10000^(2/5))]

Positional Embeddings - Rotary (RoPE)

Combines relative and absolute position

$$f_{\{q,k\}}\left(\boldsymbol{x}_m, m\right) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W^{(11)}_{\{q,k\}} & W^{(12)}_{\{q,k\}} \\ W^{(21)}_{\{q,k\}} & W^{(22)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix}$$

“rotate the affine-transformed word embedding vector by the number of angle multiples of its position index”

Relative Positional Embedding

Example: The cat sat on

Distance between tokens

The model learns how far apart tokens are

Token Pair	Relative Distance	Embedding
The - "cat"	1	[0.2, 0.5, -0.1]
cat - "The"	-1	[-0.3, 0.1, 0.4]
cat - "sat"	1	[0.2, 0.5, -0.1]
sat - "cat"	-1	[-0.3, 0.1, 0.4]
The - "sat"	2	[0.8, -0.2, 0.3]
sat - "The"	-2	[-0.7, 0.6, -0.5]
cat - "on"	2	[0.8, -0.2, 0.3]
on - "cat"	-2	[-0.7, 0.6, -0.5]
The - "on"	+3 (clipped to +2)	[0.8, -0.2, 0.3]
on - "The"	-3 (clipped to -2)	[-0.7, 0.6, -0.5]

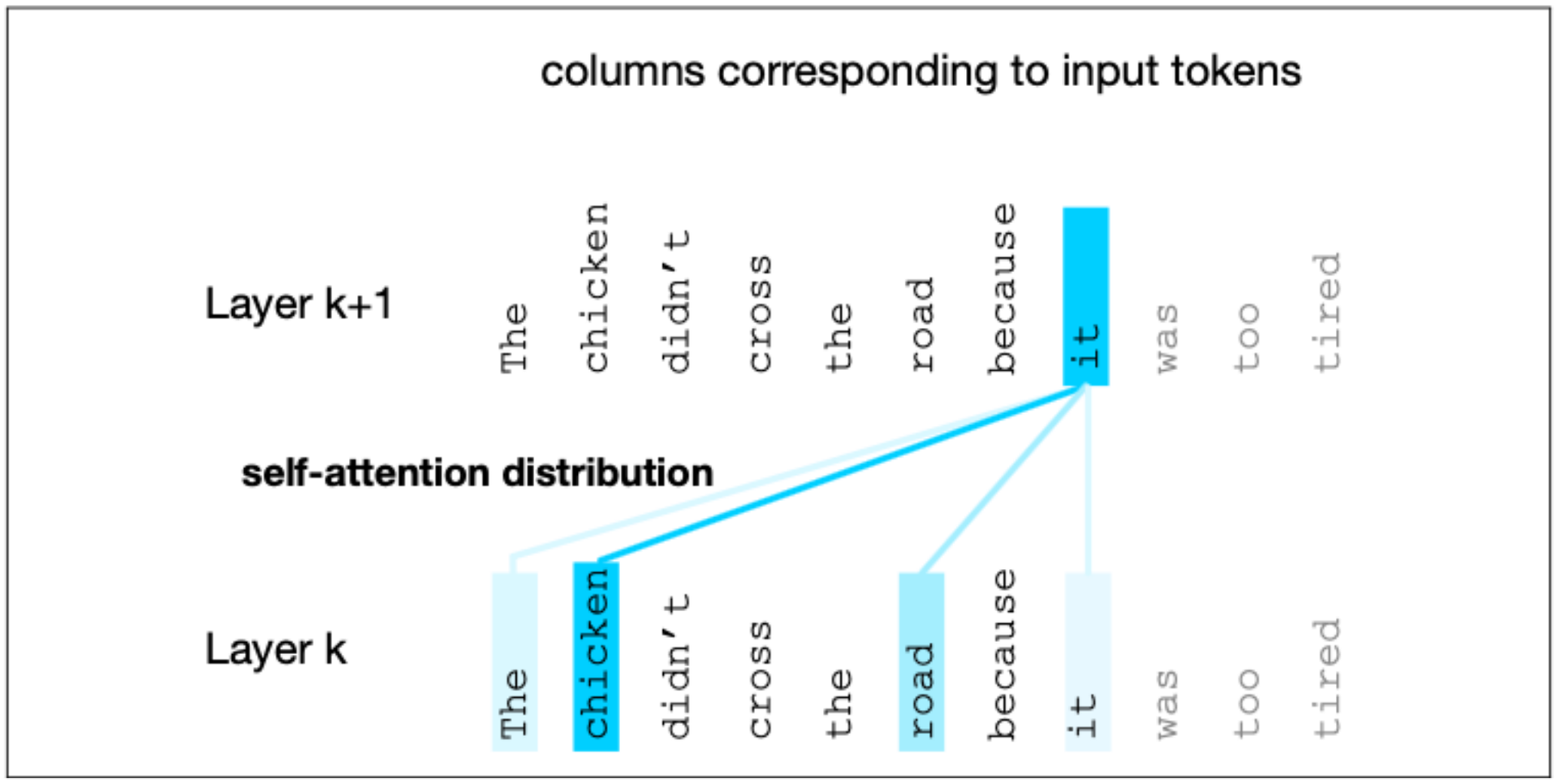
Transformers & Attention

The **chicken** didn’t cross the road because **it** was too tired.
The chicken didn’t cross the **road** because **it** was too wide.

The **keys** to the cabinet **are** on the table.
I walked along the **pond**, and noticed one of the trees along the **bank**.

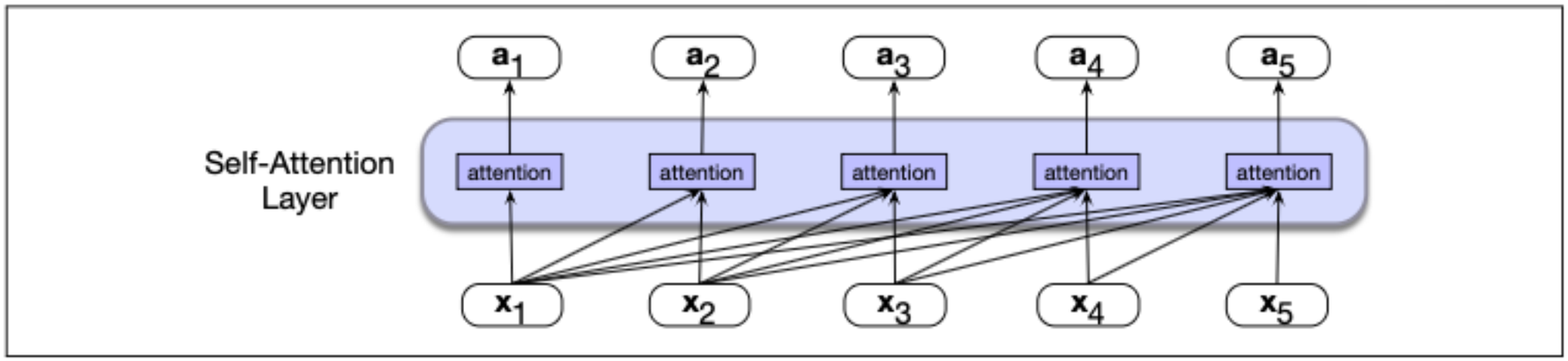
Attention

Build the relationship of tokens in layer K+1 from layer K



Speech and Language Processing, Jurafsky & Martin, Chapter 8

Simplified Attention



$\mathbf{a}_i$ Attention output at location i

$\mathbf{x}_j$

α_{ij} How much $\mathbf{x}_j$ should contribute to $\mathbf{a}_i$

$$\mathbf{a_i} = \sum_{j \leq i} \alpha_{\mathbf{ij}} \mathbf{x_j}$$

Speech and Language Processing, Jurafsky & Martin, Chapter 8

How to Compute $\mathbf{a}_i$

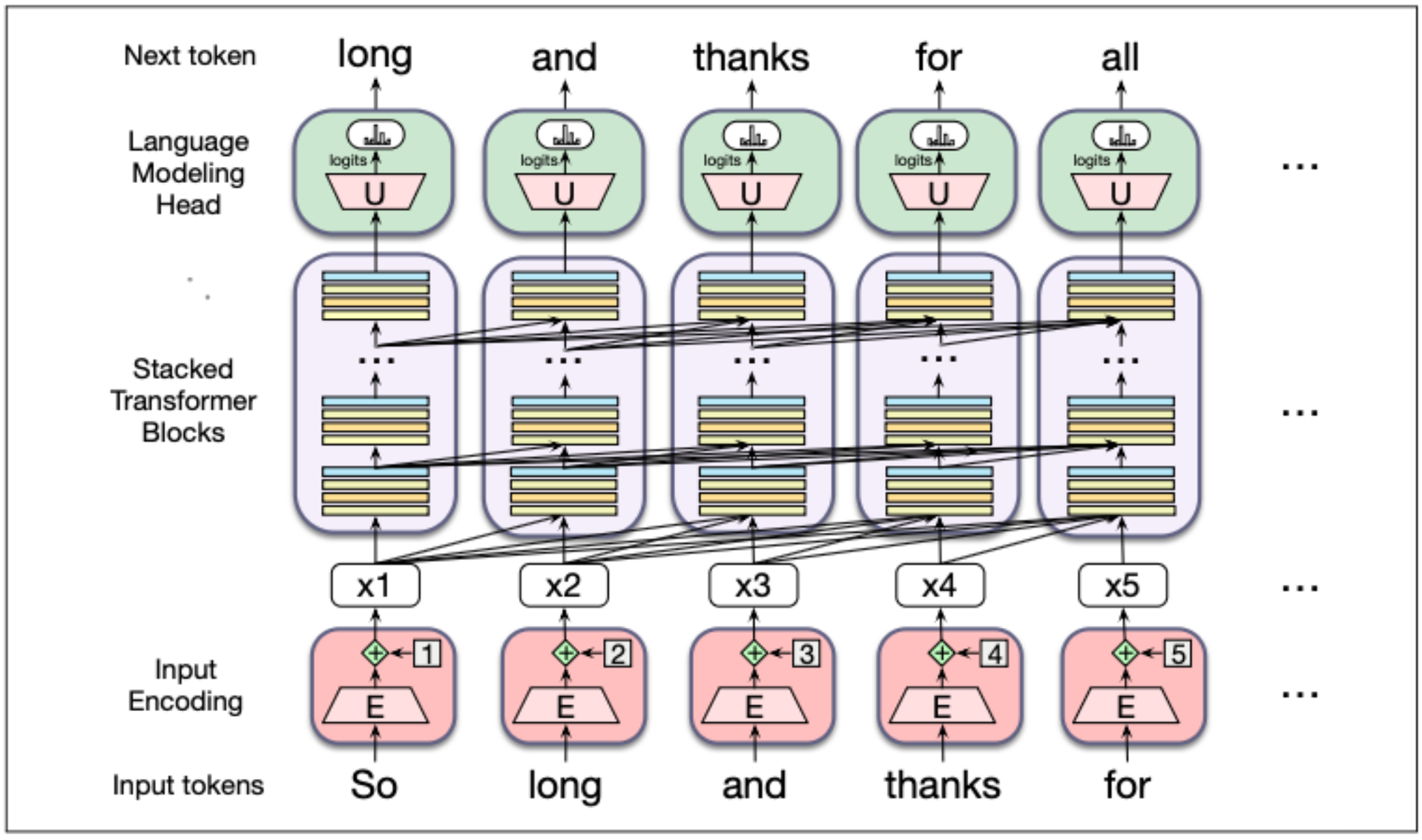
$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

$$\text{score}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \forall i \leq j$$

Real Attention

Prompt: So long and thanks for



Speech and Language Processing, Jurafsky & Martin, Chapter 8

Three Roles

query

Current element being compared to preceding elements

$\mathbf{q}$

key

A preceding element that is being compared to the current element

$\mathbf{k}$

value

The value of a preceding element that gets weighted and summed to compute the output of the current element

$\mathbf{v}$

The Calculation

Matricies to map $\mathbf{x}_i$ to query, key, value

$\mathbf{W^Q}$

$\mathbf{W^K}$

$\mathbf{W^V}$

Computing query, key, value

$$\mathbf{q}_i = \mathbf{x}_i\mathbf{W^Q}$$

$$\mathbf{k}_i = \mathbf{x}_i\mathbf{W^K}$$

$$\mathbf{v}_i = \mathbf{x}_i\mathbf{W^V}$$

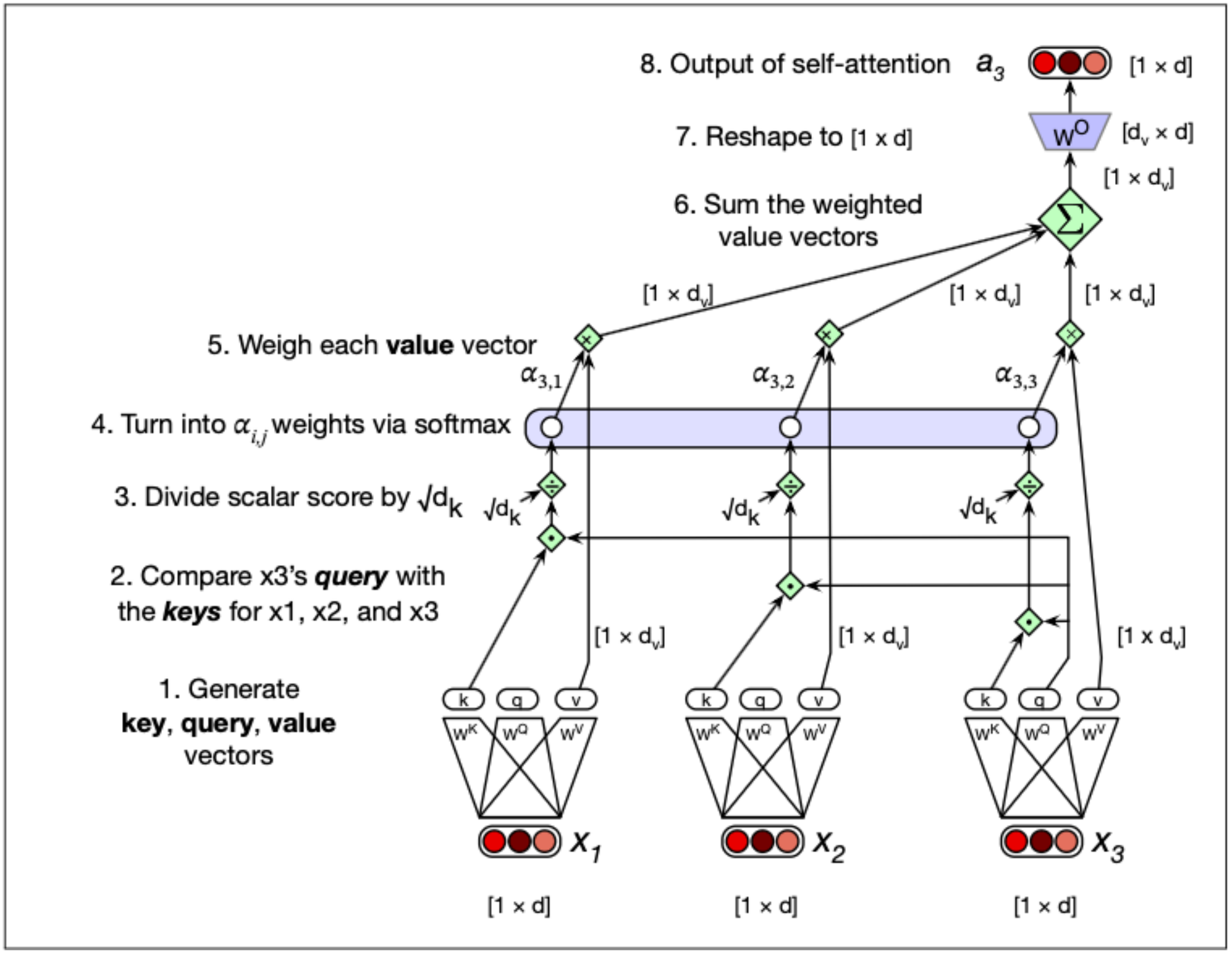
$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \forall i \leq j$$

$$\mathbf{head}_i = \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j$$

$$\mathbf{a}_i = \mathbf{head}_i\mathbf{W}^O$$

$\mathbf{W}^O$ is used to reshape the vector

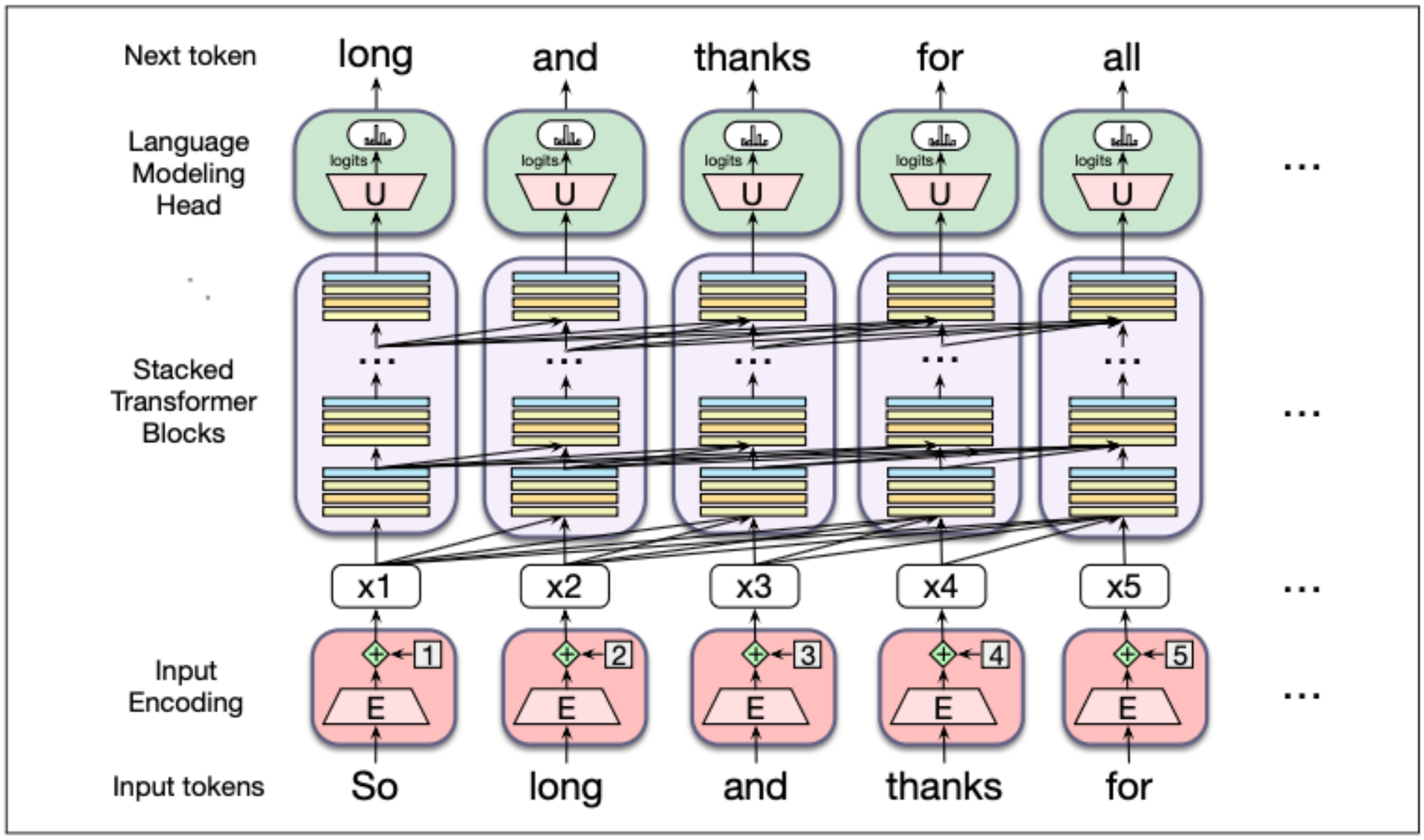


Speech and Language Processing, Jurafsky & Martin, Chapter 8

K, V Issue

To compute the token after all we submit the prompt

So long and thanks for all



KV cache

For each token in the context window, store Key and Value for every layer

$$Memory = 2 \times \text{Layers} \times d_{model} \times \text{Context Length} \times \text{Precision}$$

Llama 3 8B

- Layers: 32
- d_{model}: 4,096
- Precision: 2 bytes (BF16)

Memory requirements

- Model Weights: ~15GB
- KV cache: ~64 GB