
DAY 1 · LANGUAGE MODELS · PART 4 OF 5

BERT and GPT

Masked language models, autoregressive models — and how GPT generates text

Pavlos Protopapas

Harvard University



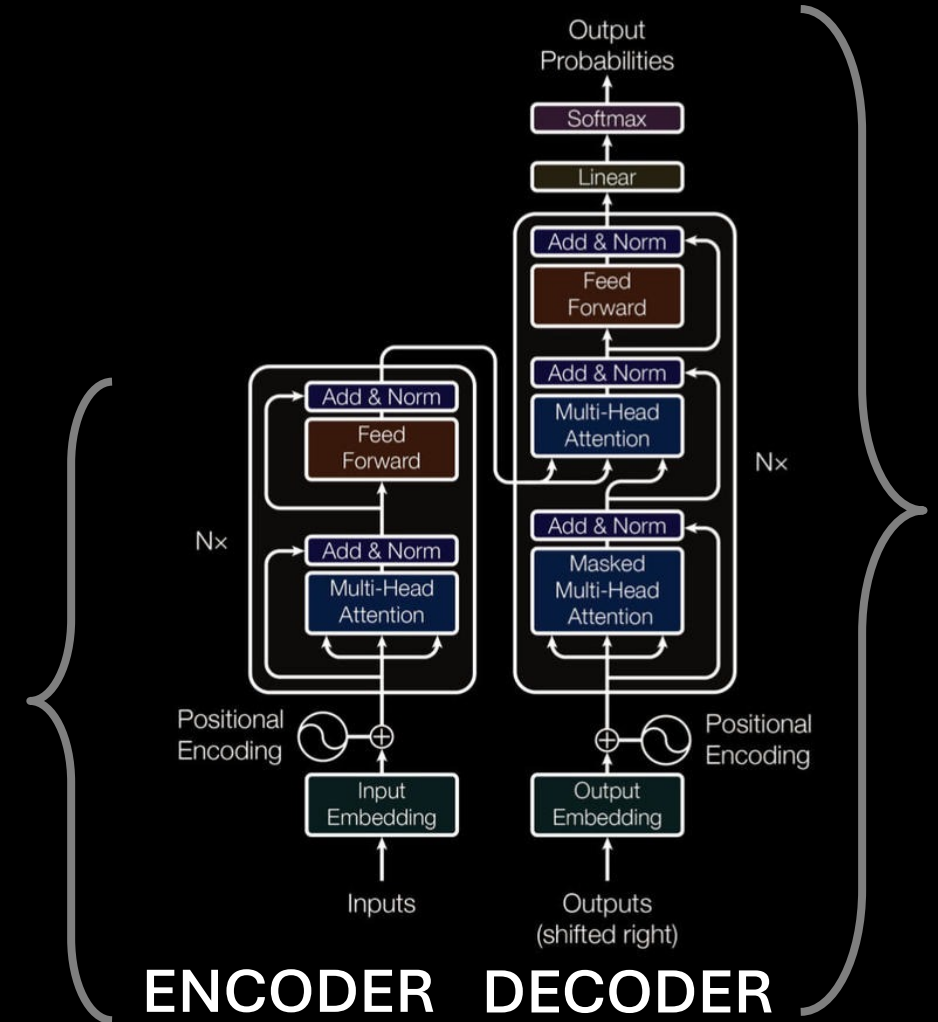
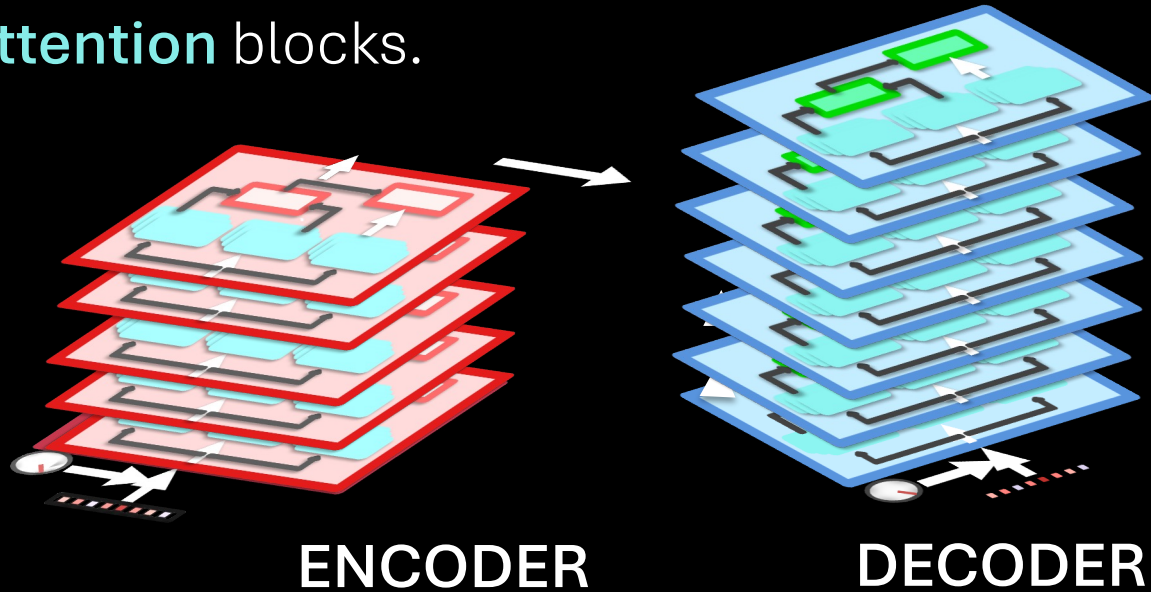
Outline

- › **Transformers Recap**
- › BERT
- › Practical Problems in Large Language Models

Recap | Transformers

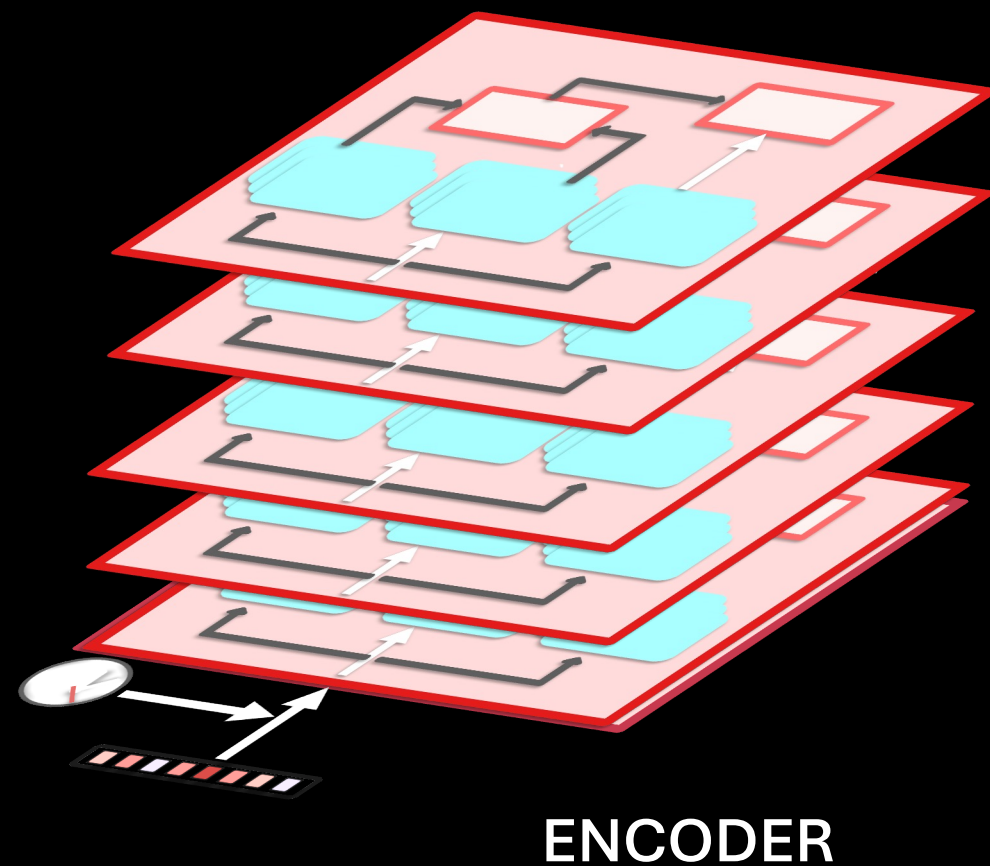
Transformers were originally devised to help with neural **machine translation**.

They consist of an **encoder-decoder architecture**, using stacked multi-head **attention** blocks.



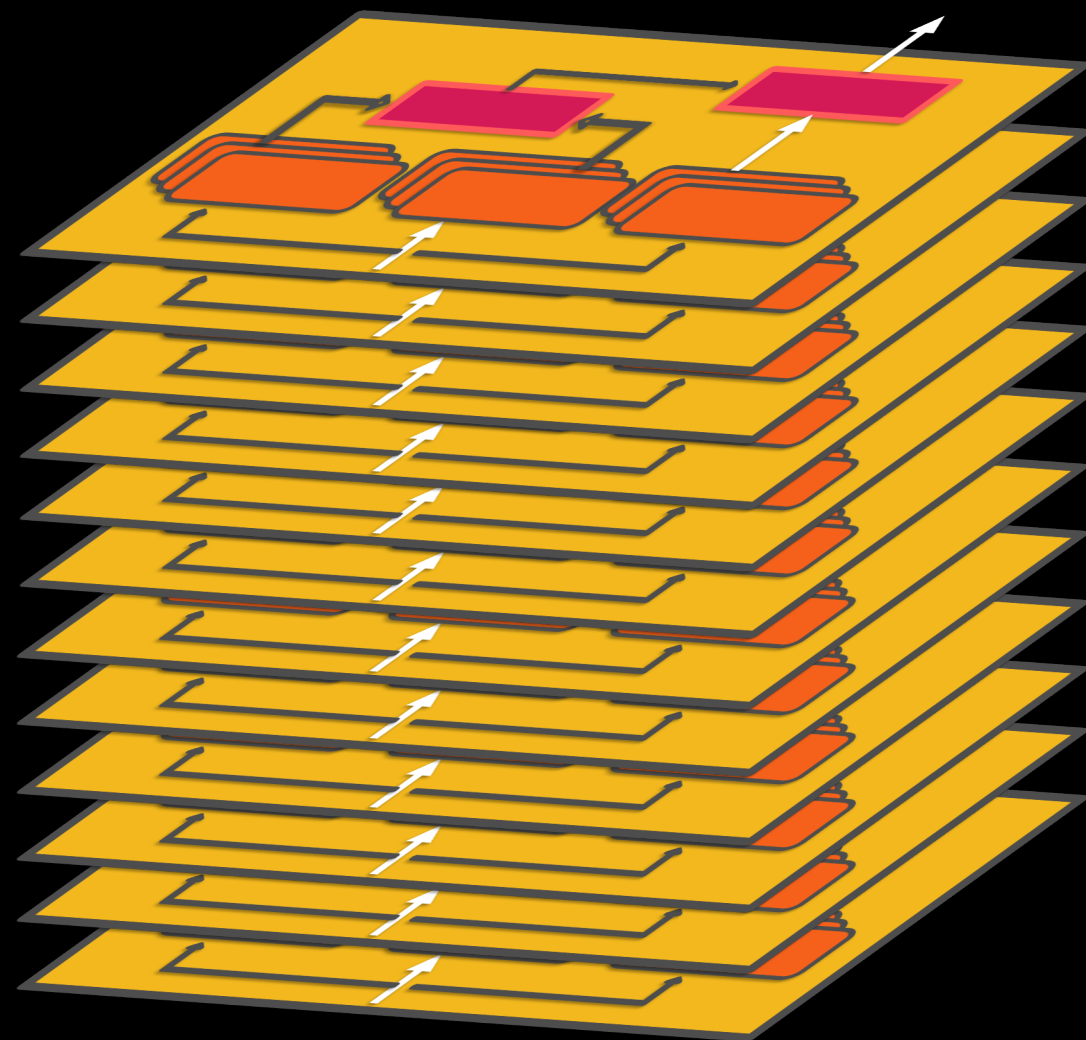
From Transformers to BERT

- Instead of an **encoder-decoder** architecture for machine translation, what if we just use the **encoder** as a language model?
- This led to a new architecture called **Bidirectional Encoder Representations from Transformers**, more commonly known as **BERT**.

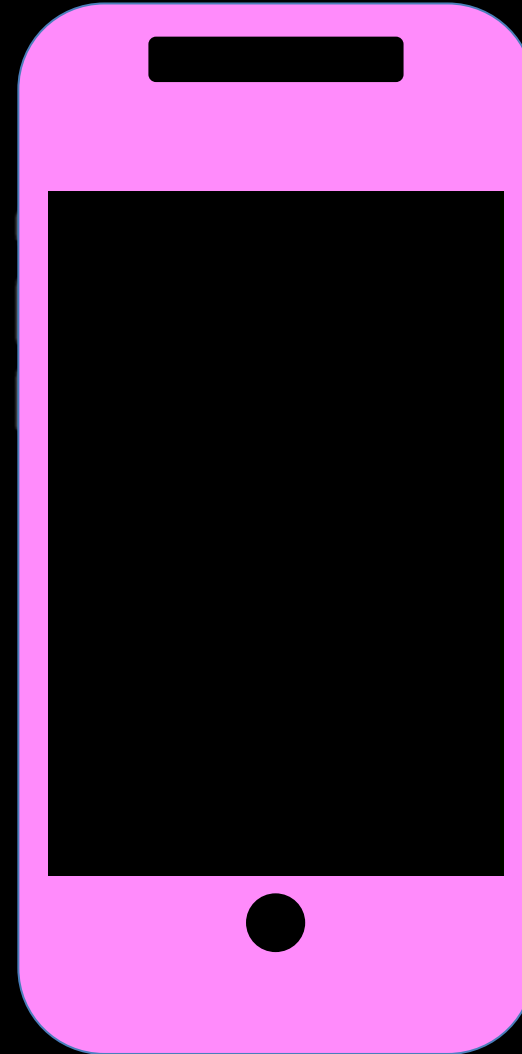
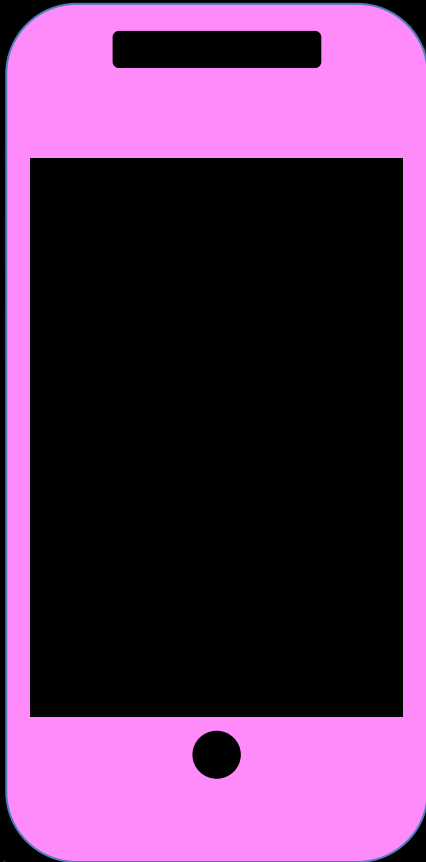


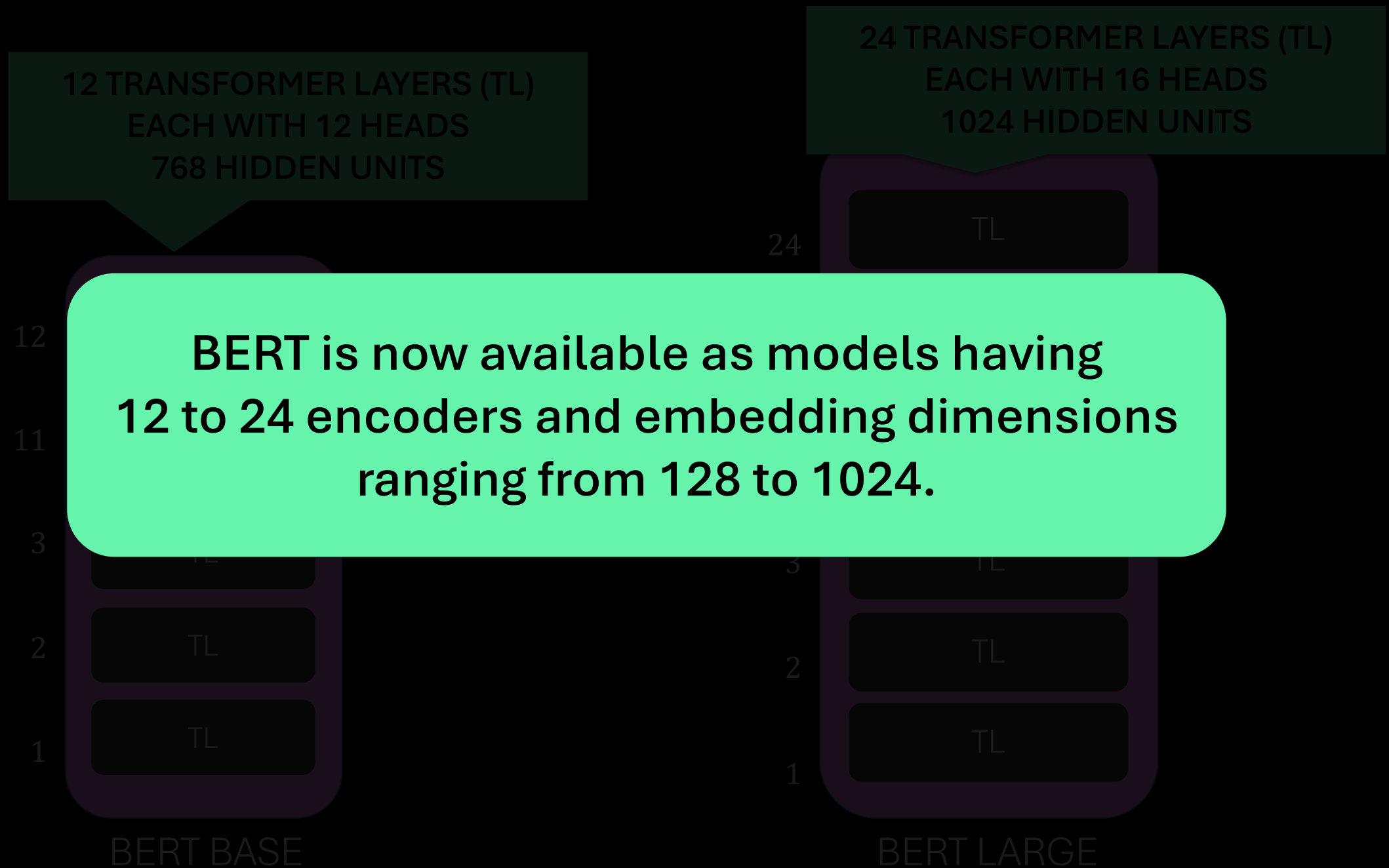
BERT

- › BERT set new records in language model tasks.
- › BERT's code and models are open-source and adaptable.
- › BERT is ideal for transfer learning.

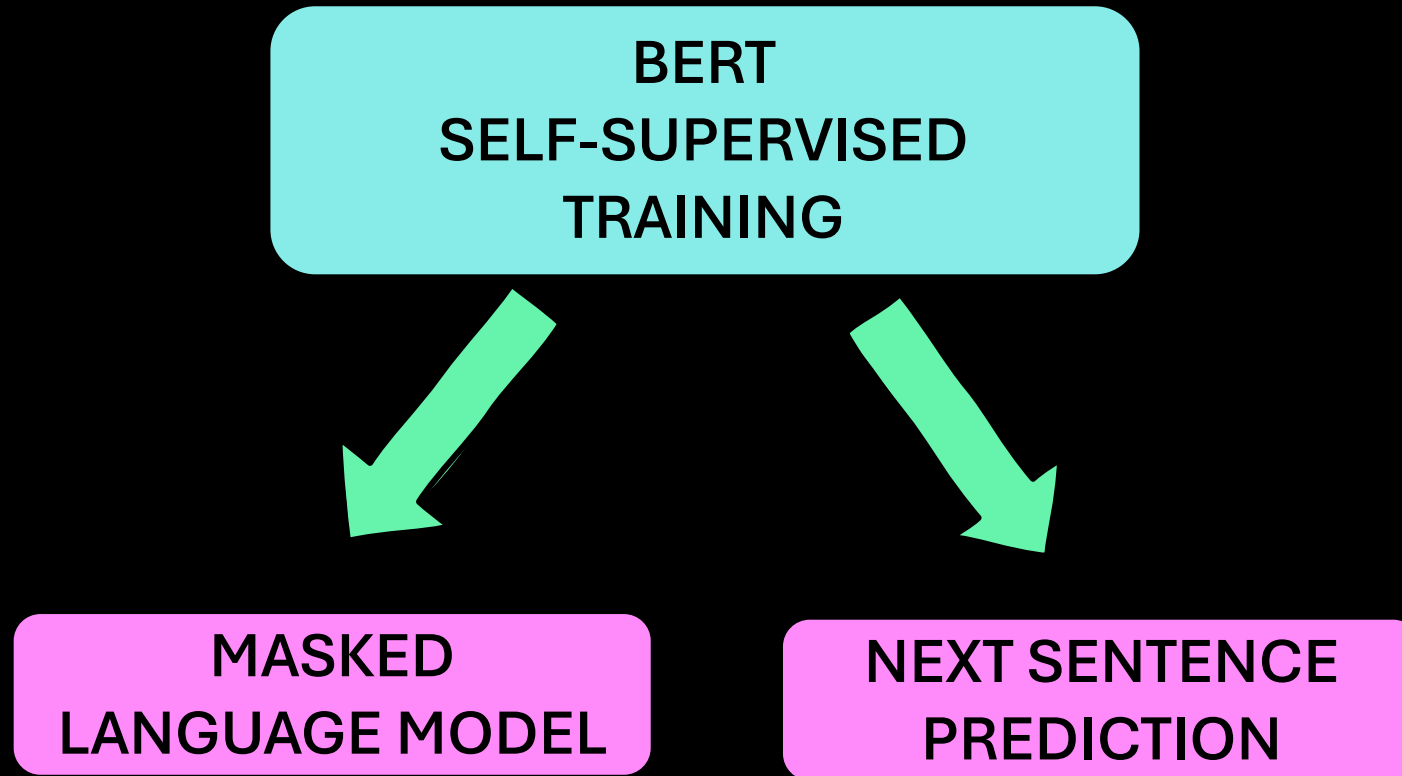


Flavors of BERT





How to Train a BERT?



RECAP | Language Model

Chester was hit by a bus as they were crossing the street.

Chester was hit by a ____



So, how do we
do this with
self-attention?

This type of language
modeling is not
possible with self-
attention because it
inputs the entire
sequence at once.



Masked Language Model

Chester was hit by a bus as they were crossing the street.

Chester was hit by a [MASK] as they were ...

Let's use my favorite
trick: MASKING!



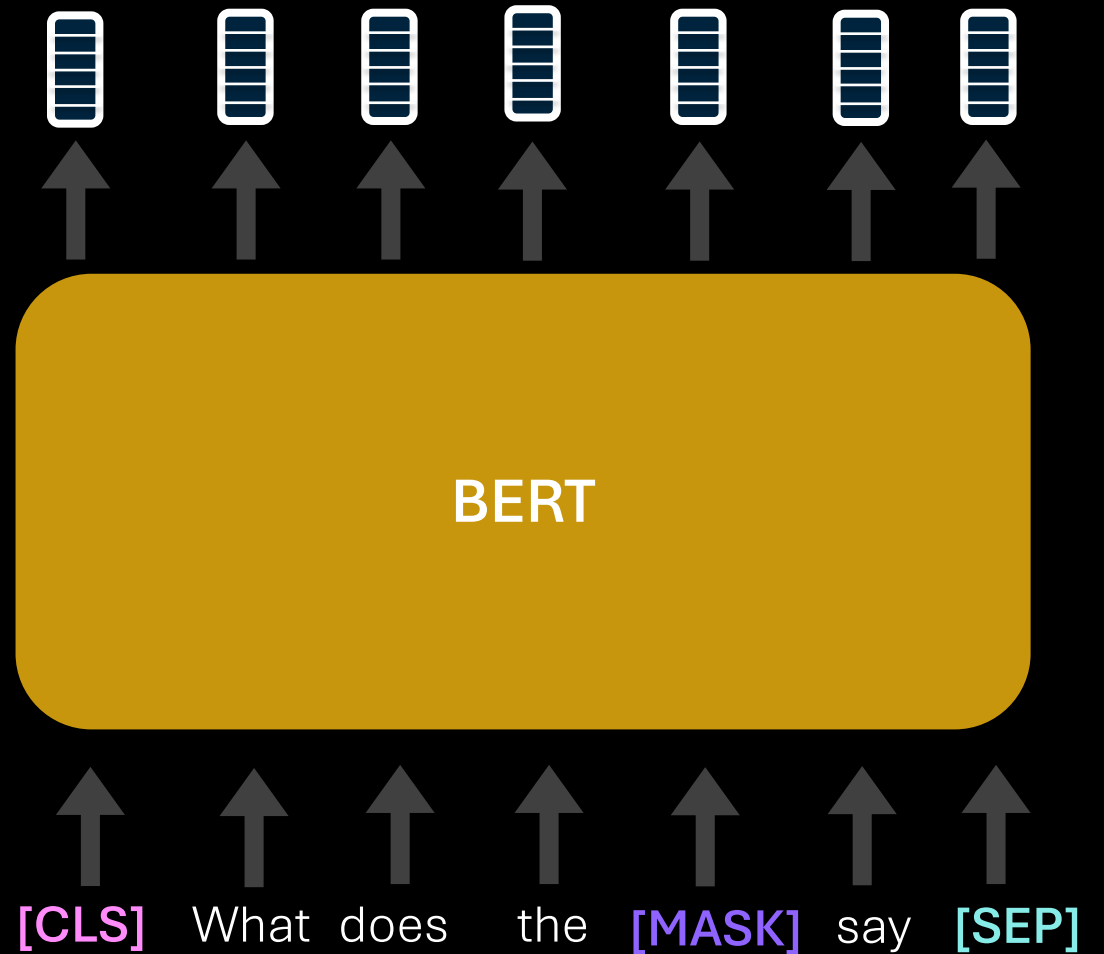
Great! We SHALL call
this a masked
language model.



BERT

TECHNICAL DETAILS

- › Input sequences to BERT must have the following special tokens:
 - › **[MASK]** – Masking token
 - › **[CLS]** – Classifier Token
 - › **[SEP]** – Separator token



Chester was hit by a bus as they were crossing the street.

This is the **top layer** which returns logits/probability distribution.

Loss function is only evaluated on masked tokens.

Dense +
softmax

$$L = -\log P(y = \text{bus})$$

0.25

bus

BERT

BERT

MASKING ISSUES?

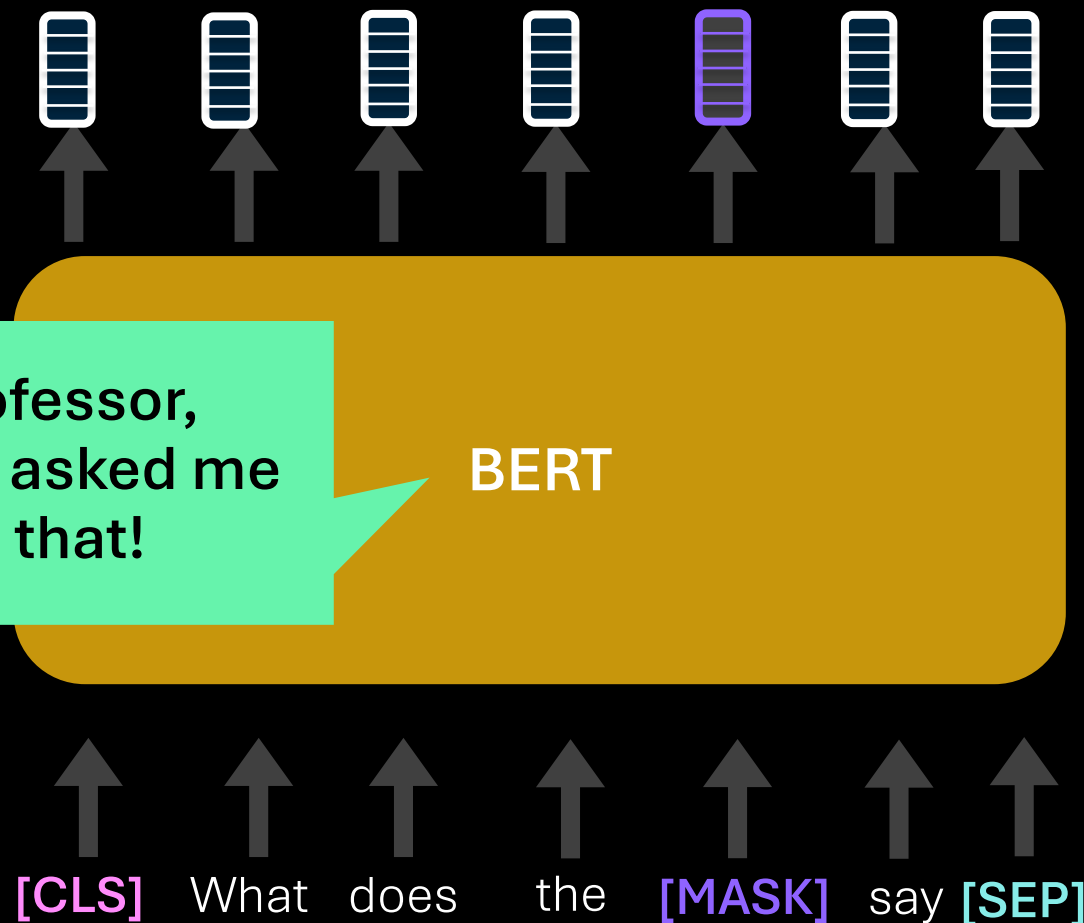
- The model learns how to correctly predict the **masked** words but completely ignores other words.



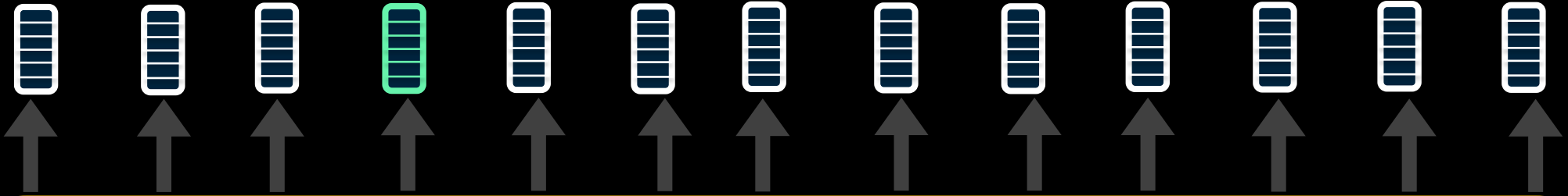
Hey BERT,
why aren't you
predicting other
words correctly?

But professor,
you never asked me
to do that!

BERT



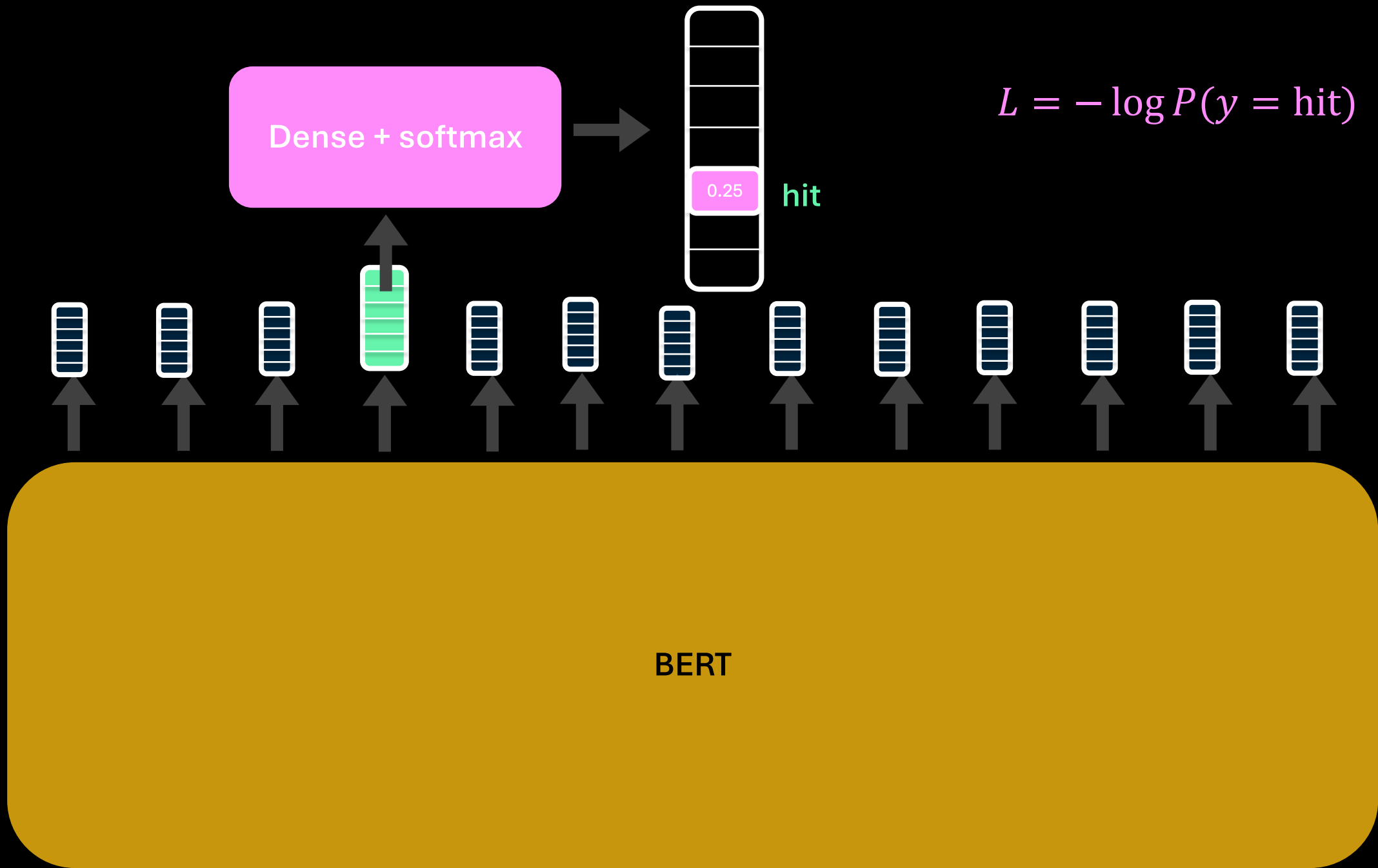
Chester was **hit** by a bus as they were crossing the street.



Occasionally, we randomly pick an **unmasked word** and ask the network to predict it.

BERT





BERT

ISSUES?

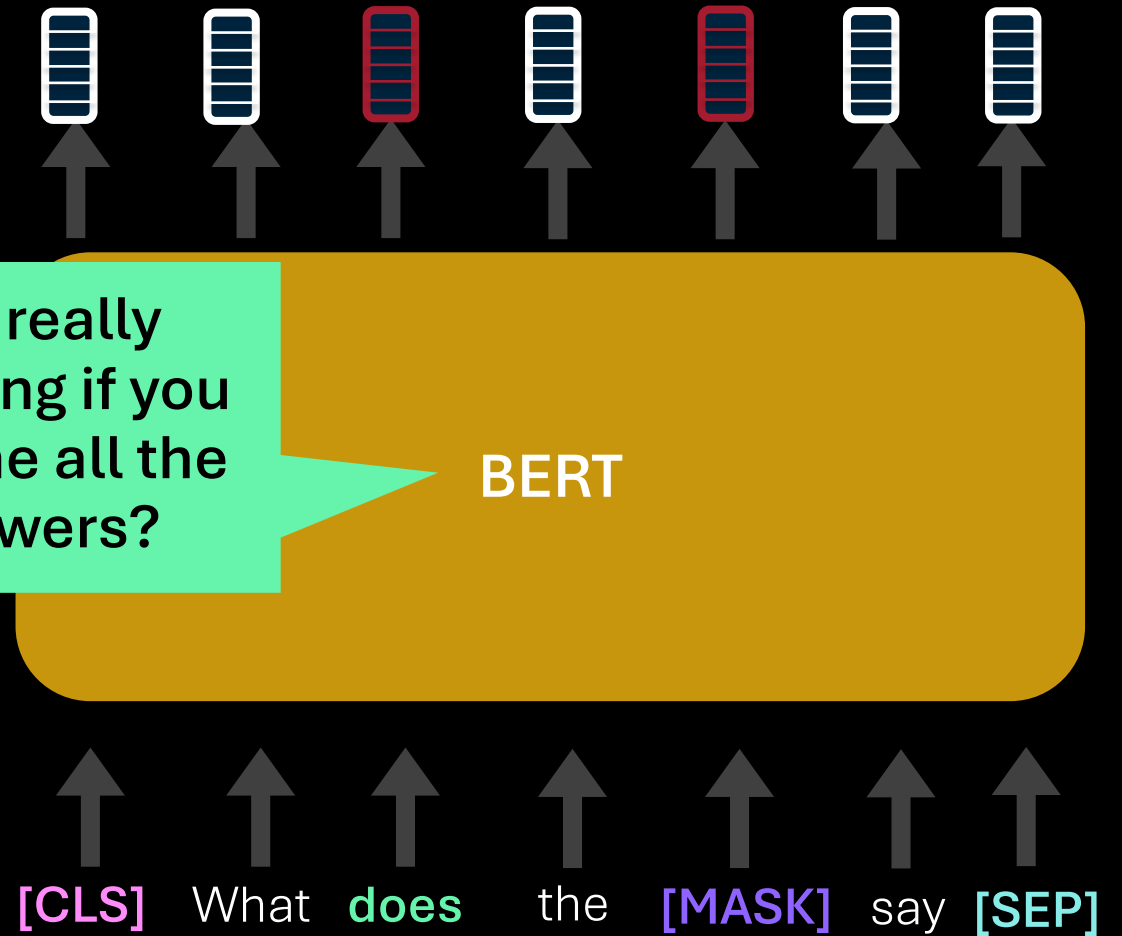
- › The model could use an identity mapping to return **unmasked** words.



Hey BERT,
why are you
cheating on the
test?

Is it really
cheating if you
give me all the
answers?

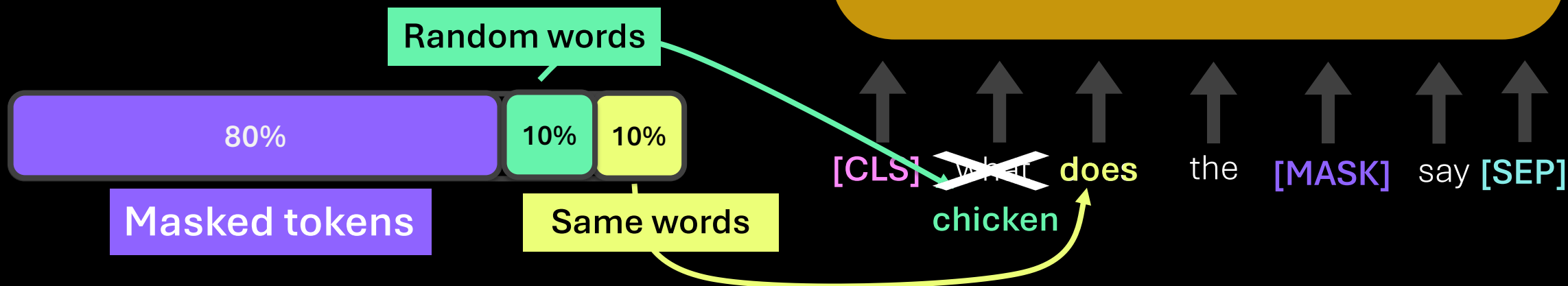
BERT



BERT

MASKING DETAILS

- › BERT's language modeling task selects **15% of the input tokens** and asks the model to predict the selected words.
- › **80% of the selected tokens are masked**, **10% are the same words**, and **10% are randomly replaced words**.

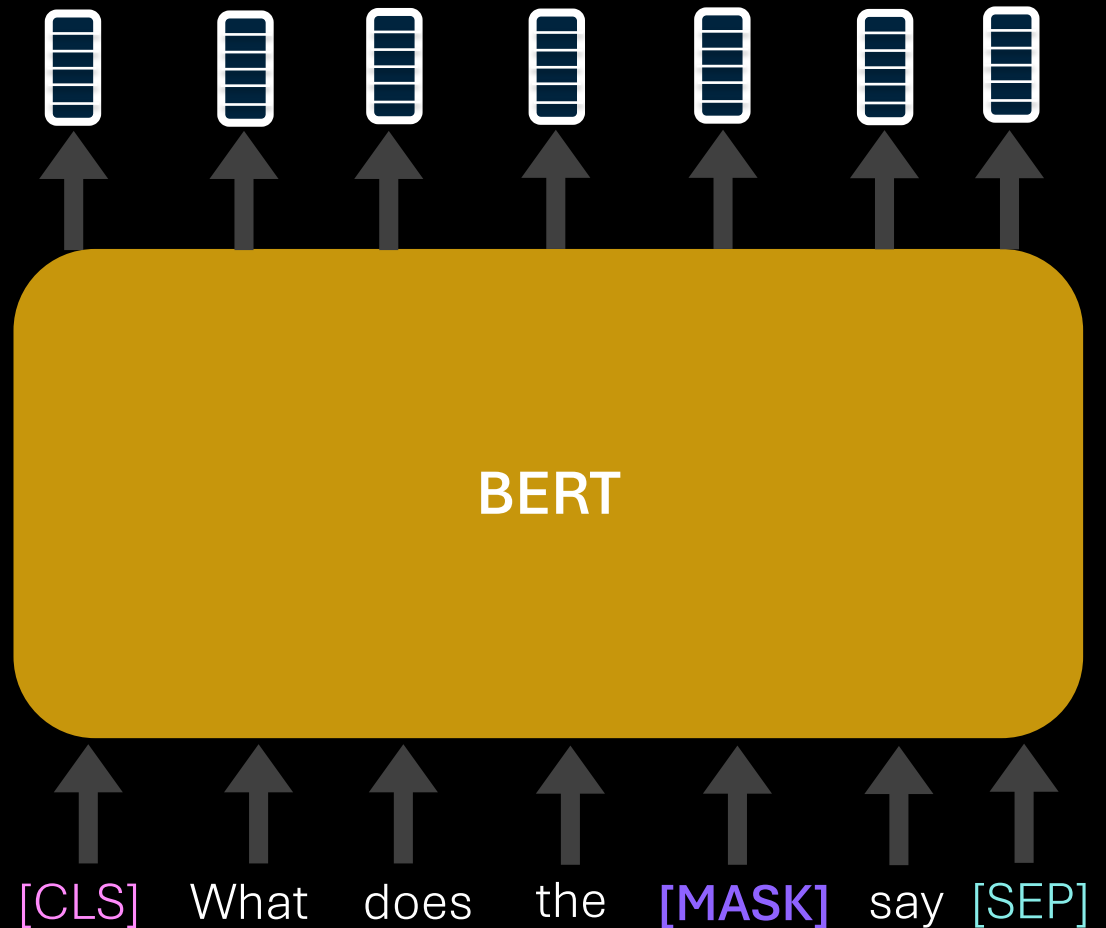


NEXT SENTENCE PREDICTION

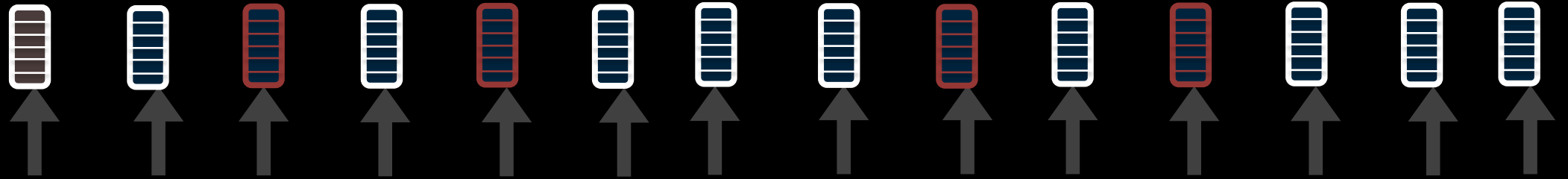
Two Sentence Task

TECHNICAL DETAILS

- › To make BERT better at handling **relationships** between **multiple sentences**, the training process includes an additional task:
 - › Given **two sentences (A and B)**, is B likely to be the sentence that **follows** A, or not?
 - › The individual sentences are separated by the **[SEP]** tag mentioned before.

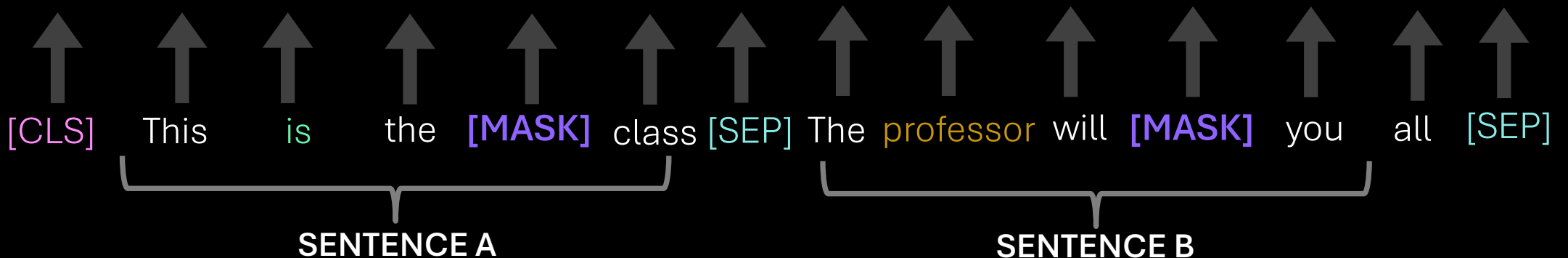


This is the **last** class. The professor will **miss** you all.

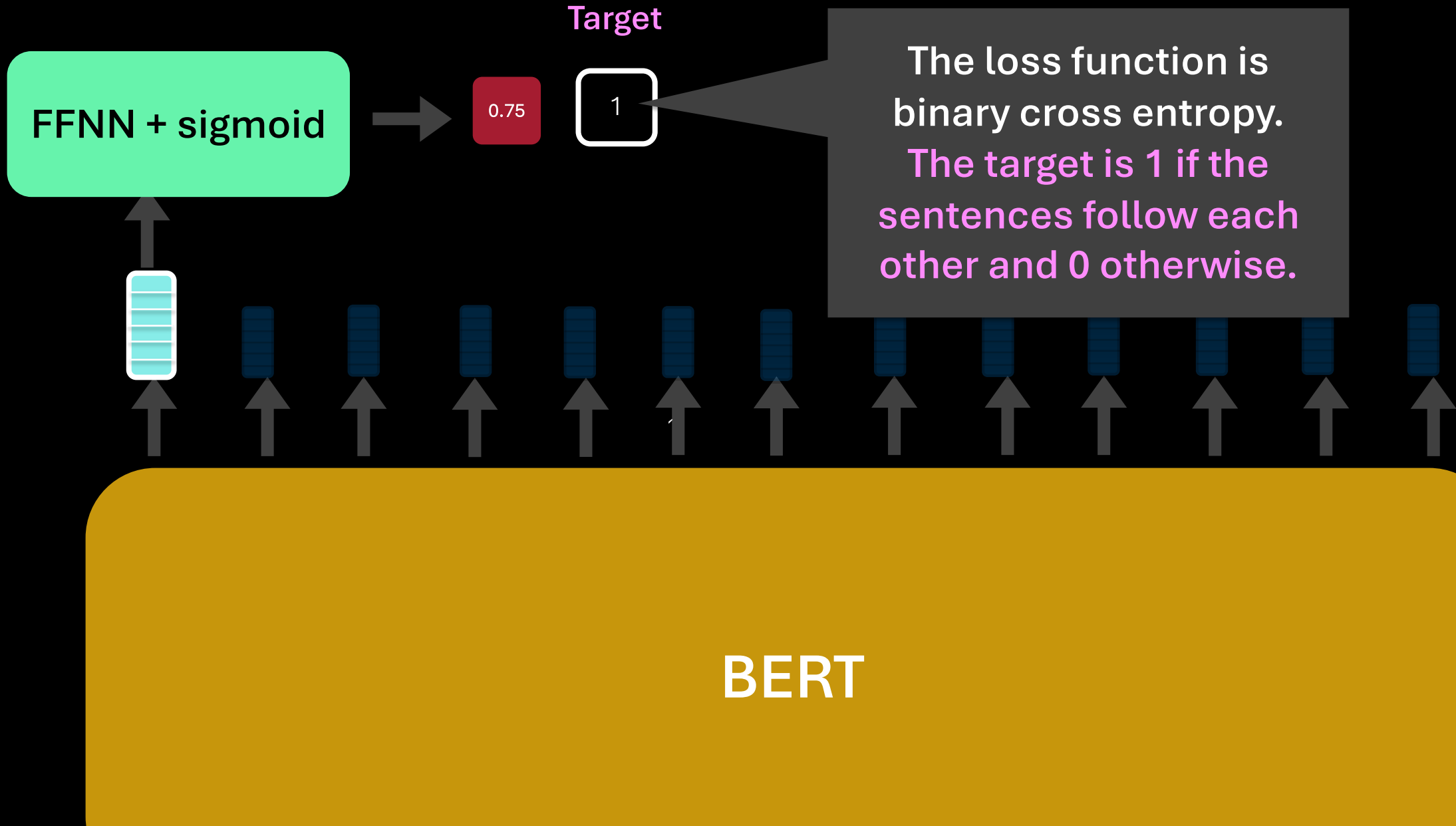


To assess if the two sentences follow each other, we use the embedding of the **[CLS]** tag from the start of the sentence.

BERT



This is the **last** class. The professor will **miss** you all.



Two Sentence Task



Do you notice any
issues here?

The texts used for training have
continuous sentences that
follow each other.
Therefore, all of the examples for
training would be labeled 1!



Negative Sampling

However, this **understanding** is **not** intuitive for a language model.

- We need to create ‘**synthetic**’ **negative samples** of sentences that don’t belong together to ‘teach’ the model.
- We can use **sample** sentences from widely different sources and train a simple classifier.

Sentence A	Sentence B	Target
Pavlos is a cheese aficionado.	Cheese makes every meal extraordinary.	1
Pavlos is a cheese aficionado.	Brown butter is the secret ingredient.	0
Brown butter is the secret ingredient	Cheese comes in many varieties.	0
Brown butter is the secret ingredient.	It adds a rich, nutty flavor.	1

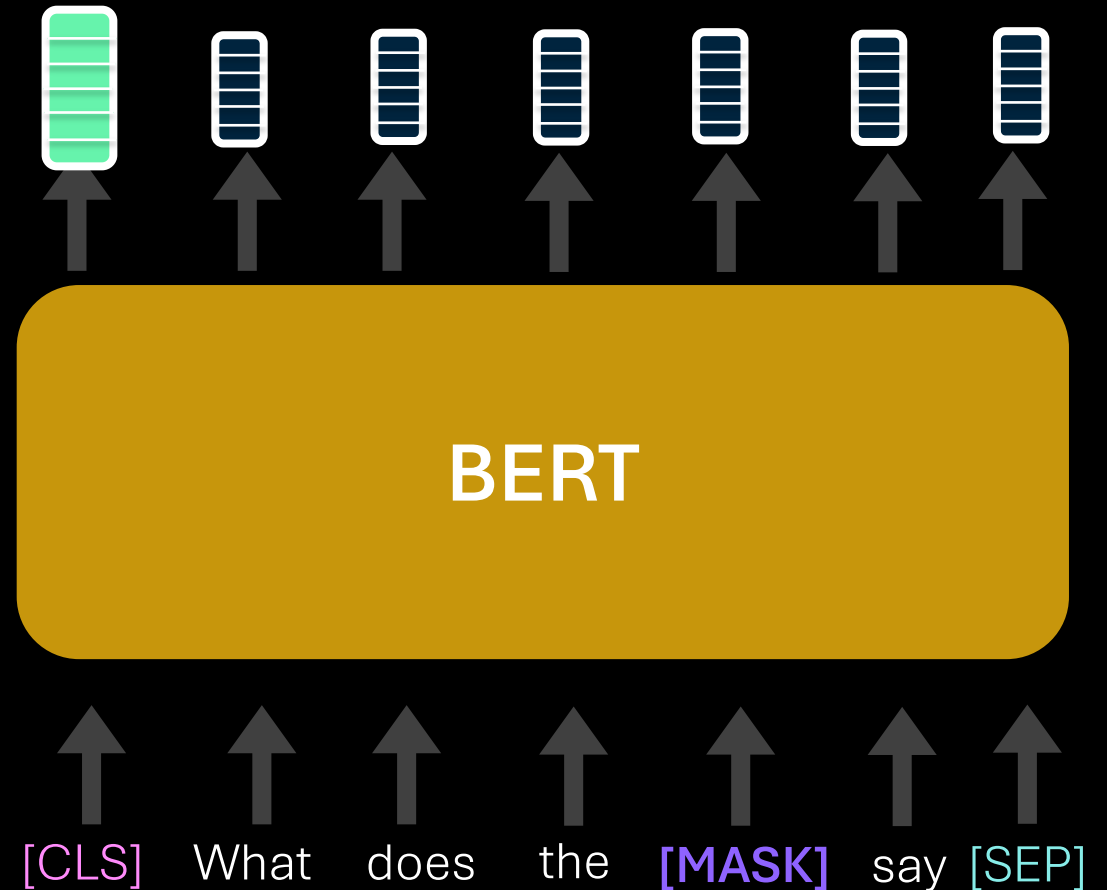
RECAP | How to Train a BERT?



How do you
train a BERT?

SUMMARY

- › Using a large corpus (e.g. Wikipedia), we pre-train a BERT model for two tasks:
 - › **Masked word** prediction
 - › **Next sentence** prediction
- › The goal is minimizing the combined loss function.



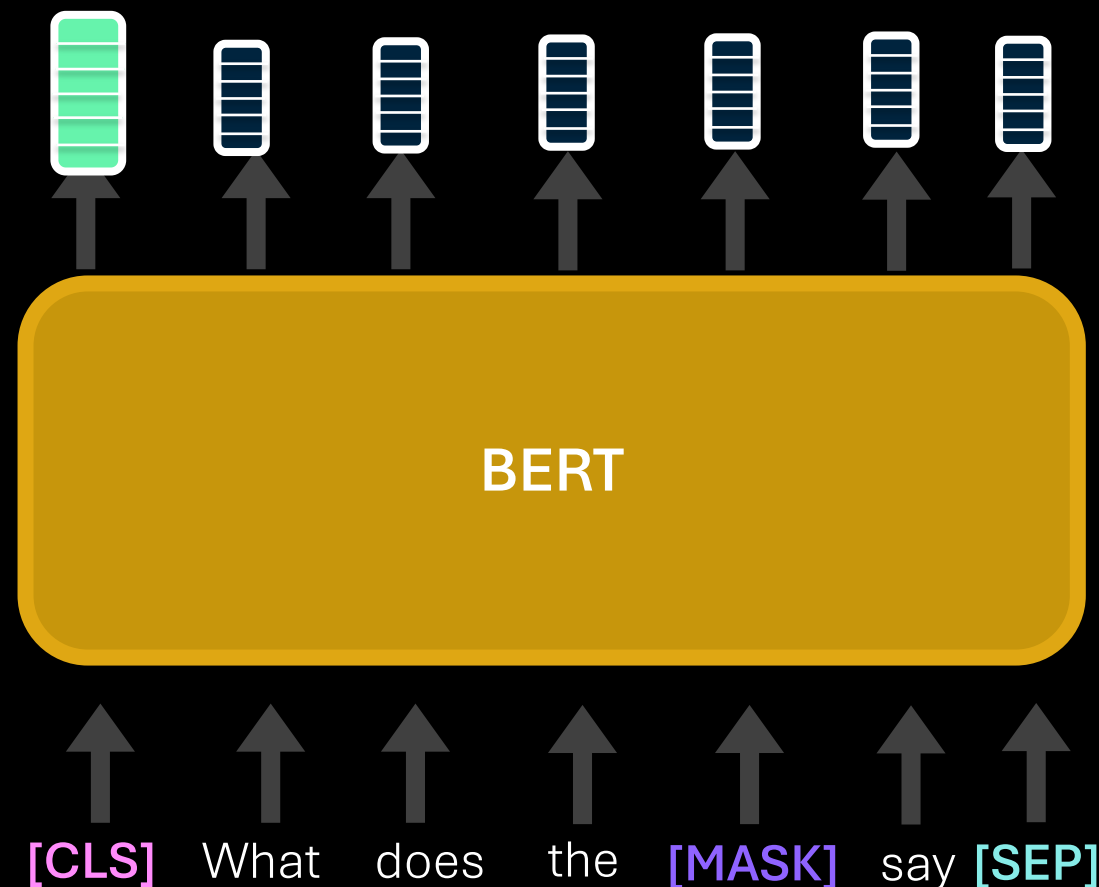
BERT Summary

› Bidirectional Encoder Architecture:

BERT reads words in both directions in a sentence, helping it better understand the order and meaning of the words.

› The Self-Attention Mechanism:

This improves how words are understood by emphasizing important information and the connections between them as the model processes the words.



BERT Issues

- › The vanilla BERT (base) has **109,482,240** trainable parameters.
- › An extremely **large** dataset is required to train such a model sufficiently.
- › The sheer size makes it extremely **slow** to **train**.
- › **Fine-tuning** —we will explain this later— requires extensive **tweaks** and **experimentation** (e.g., you must use the **same** tokenizer used during pre-training).



Vanilla BERT cannot be used for natural text generation (GPT can be used instead).

How to use a BERT?

Using BERT for a given NLP task

Pre-training



Fine-tuning



Language tasks



Pre-Training Issues

Large Dataset

Uncompressed, the total amount of data is roughly **20 to 30 GB**. That amount of storage is not unattainable, but it does require an investment to store that much data.

Computational Complexity

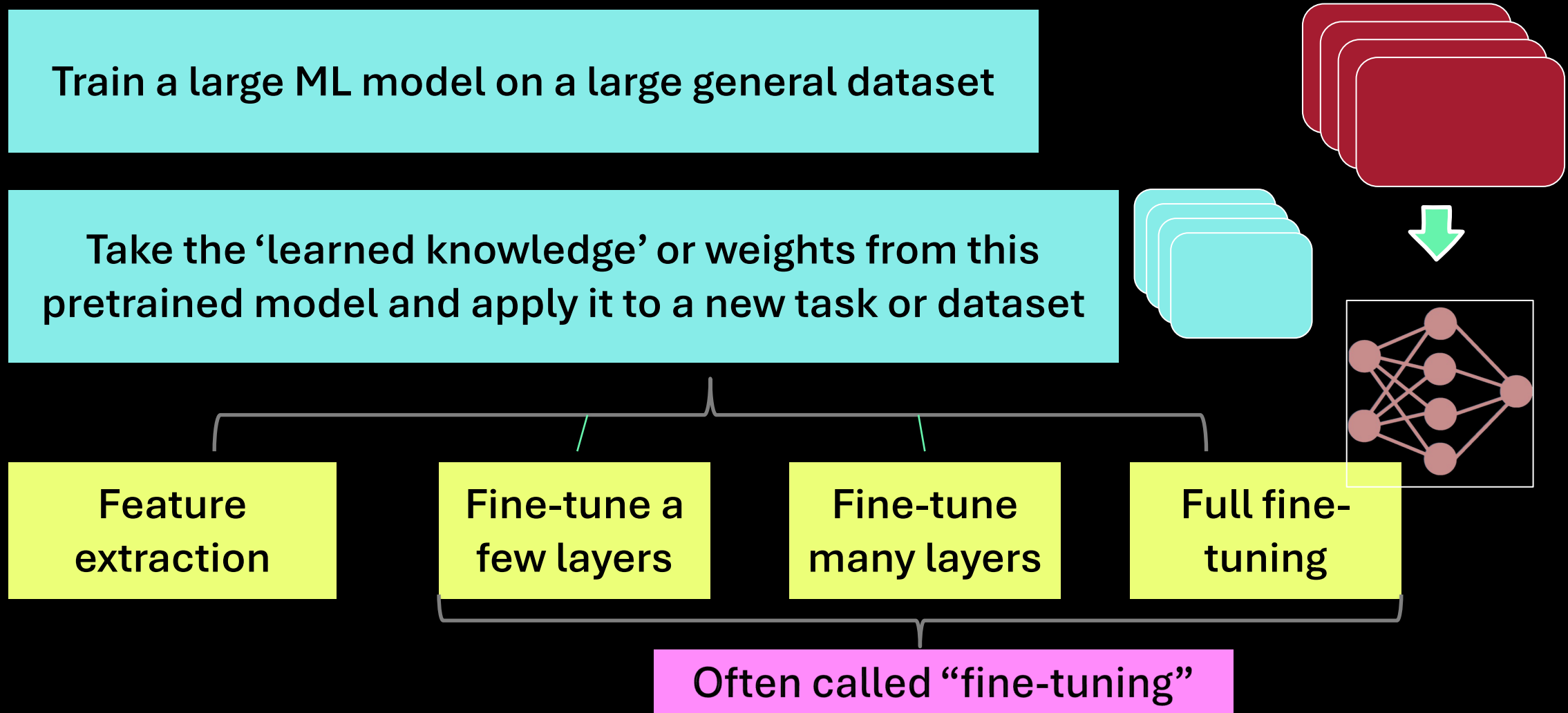
BERT was trained for 40 epochs on 16 Cloud TPUs (64 chips).

This pre-training takes around **4 days** to complete and impacts the **environment!**

Assuming you don't have that hardware or those resources at your disposal, the next solution is to use cloud resources. At a cost of USD 12 per hour, based on today's prices, this would cost around **USD 18,000**.

Basic Idea of Transfer Learning

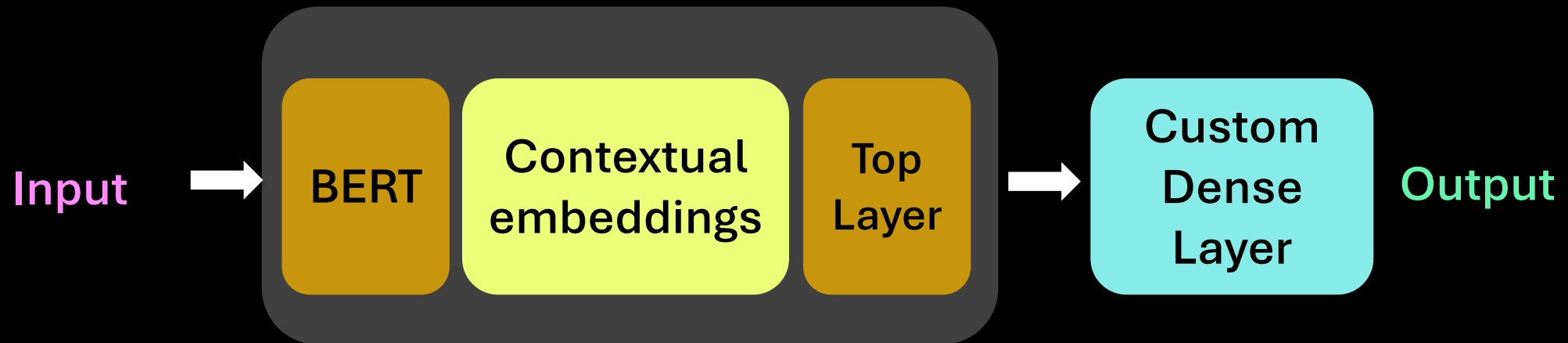
- Transfer learning is a common practice in the Deep Learning space.



How to Use BERT for NLP Tasks?

BERT can be used for a wide variety of language tasks.

- To adapt BERT for various tasks, we **first remove** its **top layer**, extract the contextual embeddings, and pass them through a **new dense layer**.
- This new layer is then trained specifically for the desired task.



How to Fine-Tune?

- ▶ We do **Masked Language Modeling** to fine-tune BERT.
- ▶ This is similar to what we do when we pretrain the model. The only **difference** is that we **load pretrained weights** trained on a very large corpus before fine-tuning the model on a domain specific dataset.
 - ▶ For a **movie dataset**:

		This is a great deal.				This is a great movie.
This is a great	→	This is a great success.		This is a great	→	This is a great film.
[MASK].		This is a great adventure.		[MASK].		This is a great story.
		This is a great idea.				This is a great character.
Before Fine Tuning				After Fine Tuning		

BERT | Key Takeaways

- › **Bidirectional** Transformer Encoder, pre-trained with masked language modeling and next-sentence prediction.
- › **Start from a pre-trained model**—you almost never train one from scratch.
- › **Adapt it to a task** with a small head, on labeled data—freeze the encoder or fine-tune it.
- › **Adapt to your domain** first, by fine-tuning the language model on unlabeled domain text.

DAY 1 · LANGUAGE MODELS · PART 4 OF 5

GPT



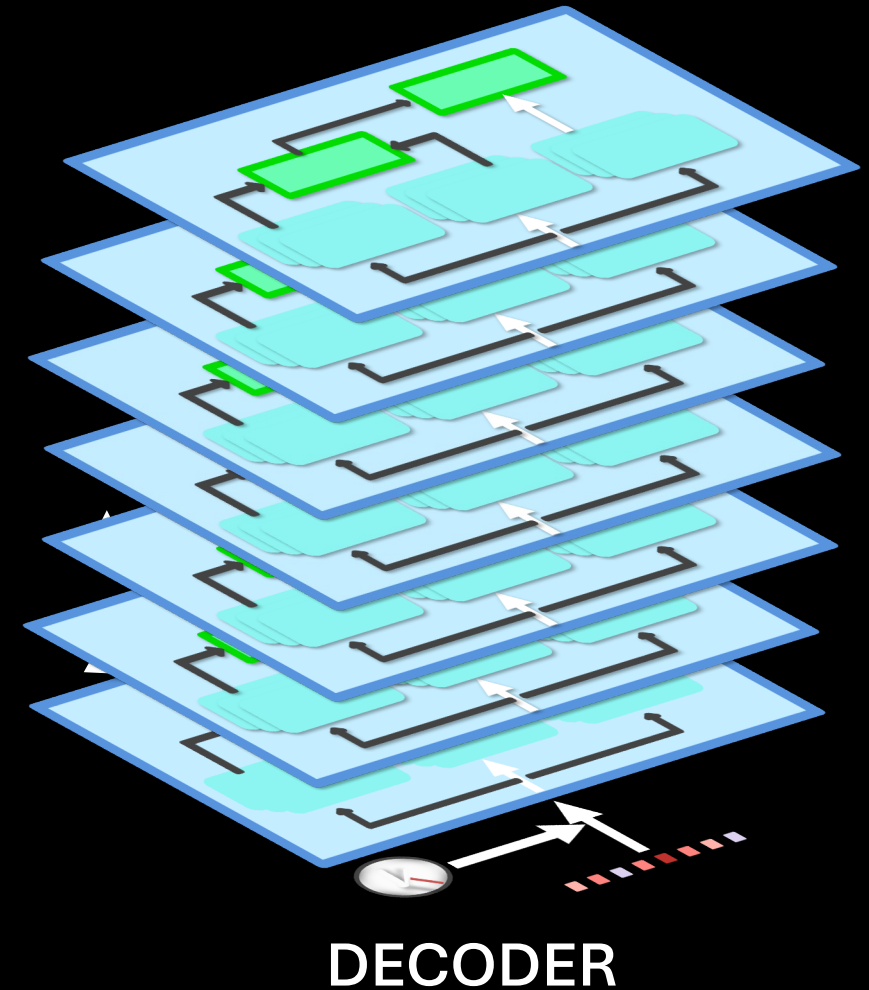
Outline

- › From Transformers to GPT
 - › Bert vs. GPT: Key Differences
 - › Autoregressive Generation
- › Inside GPT
 - › Masked Self-Attention
 - › Training GPT
 - › Layer Normalization
- › Generating Text
 - › Decoding Strategies and Beam Search
- › **GPT in Practice**
 - › The Evolution of GPT

From Transformers to GPT

- We saw how we can make use of the encoder blocks of the Transformers in BERT. Now, what if we use just the decoder blocks?
- Voilà: the **Generative Pre-trained Transformer!** Or GPT in short.

Note that we are simply using a decoder-like architecture. It is not essentially a decoder anymore!



BERT vs. GPT | Key Difference

Apart from the fact that **GPT** uses **decoder-like** blocks of the Transformer, while **BERT** uses **encoder**-like blocks, there is also a fundamental difference between the objectives of the two models:

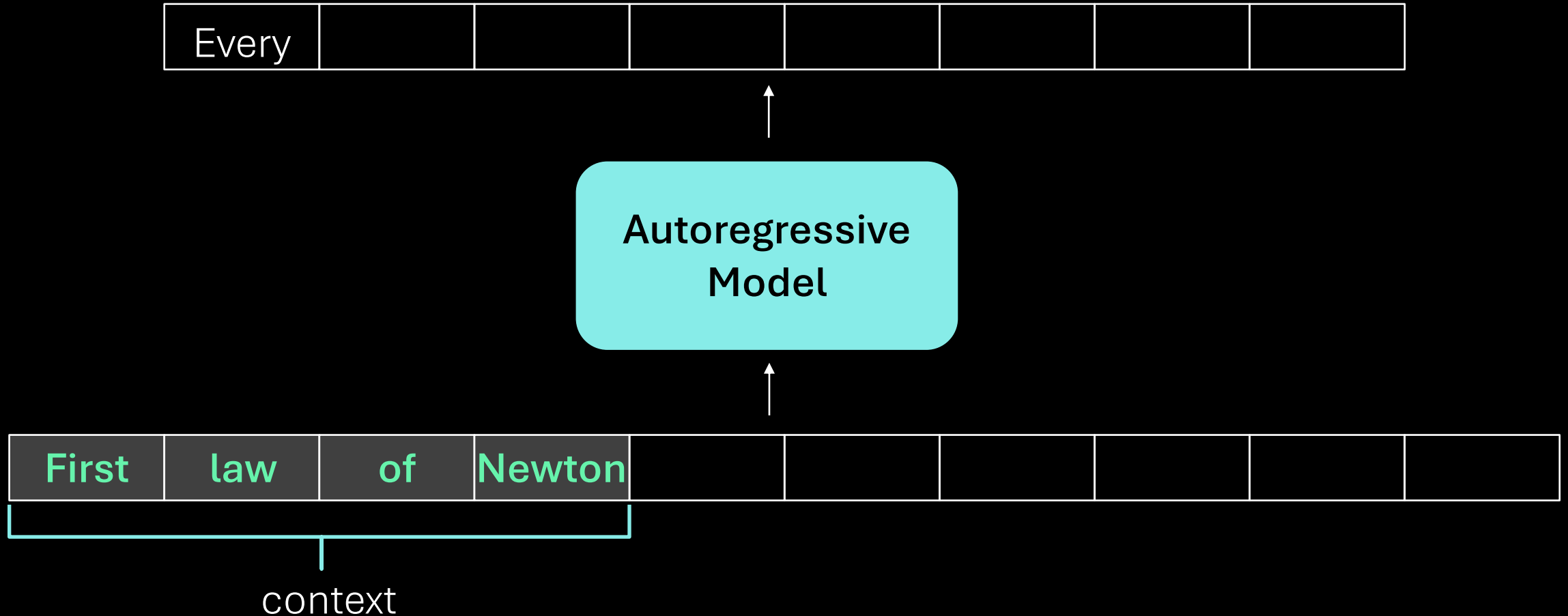
- › **BERT** is a **denoising autoencoder** model.
- › **GPT** is an **autoregressive** model.

BERT vs. GPT | Key Difference

- › **GPT** is an **autoregressive** model.
 - › An **autoregressive** model predicts the next value in a sequence. In our case, these are language tokens.
 - › Uses “**causal**” attention (sees only past words).
 - › Can generate fluent text step-by-step.
 - › Commonly used for chatbots, story writing, etc.

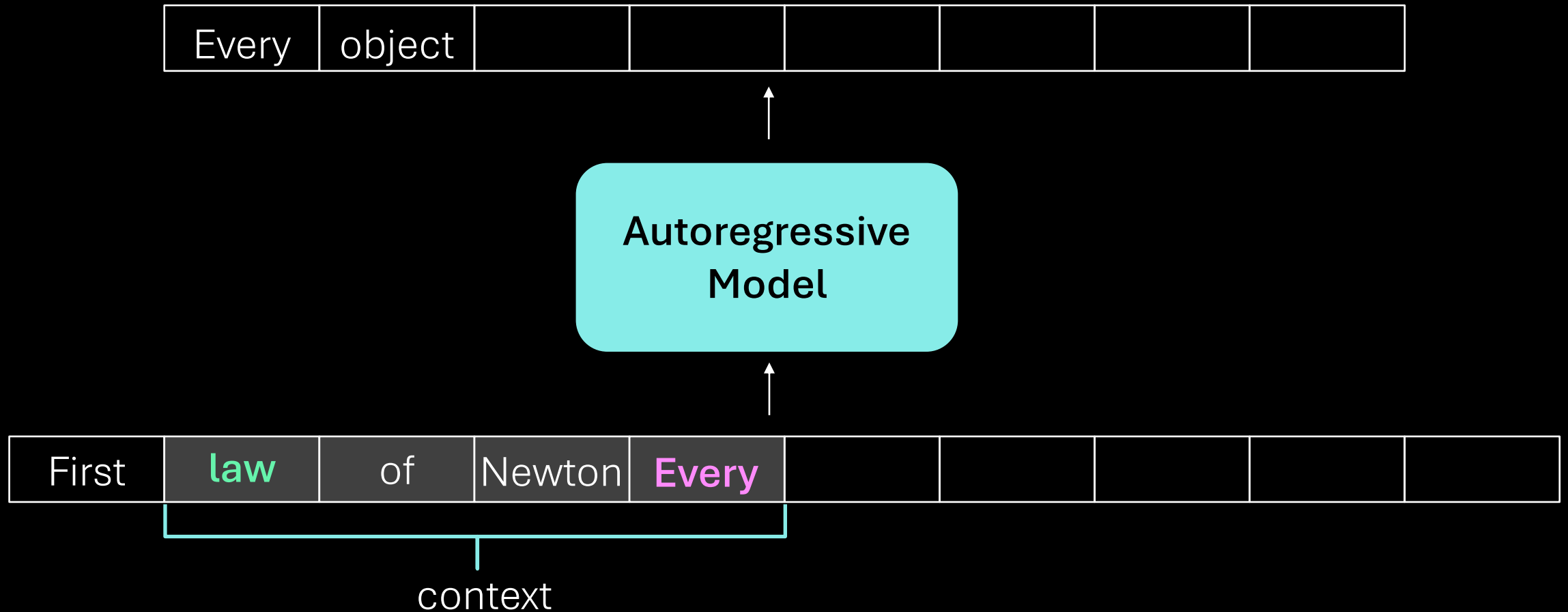
Autoregressive Model

Unidirectional processing and a fixed-length sliding window are characteristic of a general autoregressive model.



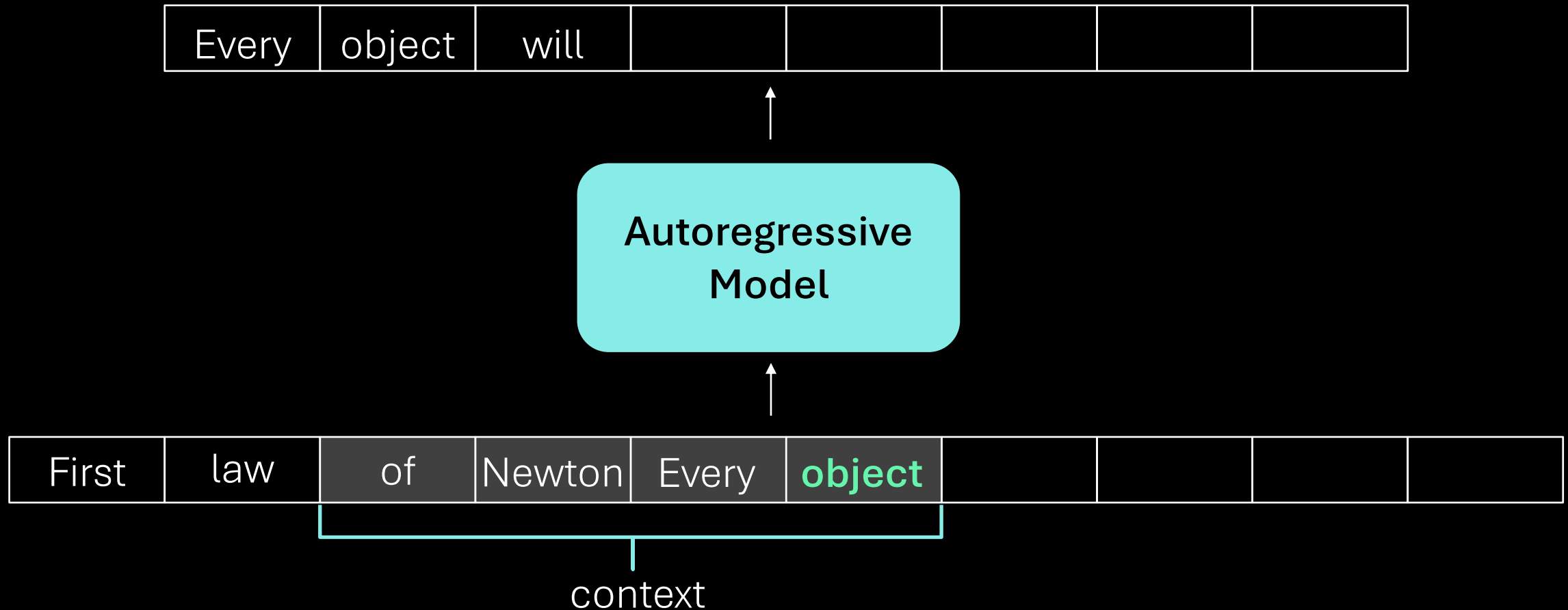
Autoregressive Model

Unidirectional processing and a fixed-length sliding window are characteristic of a general autoregressive model.



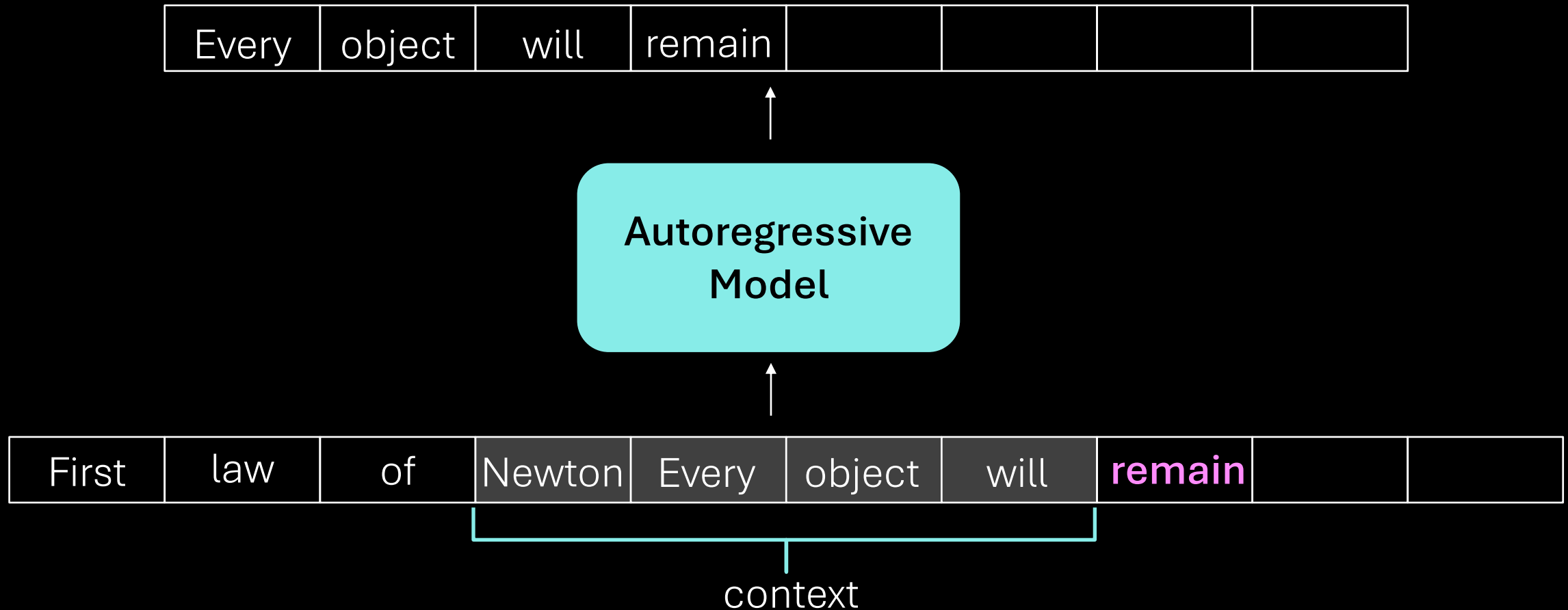
Autoregressive Model

Unidirectional processing and a fixed-length sliding window are characteristic of a general autoregressive model.



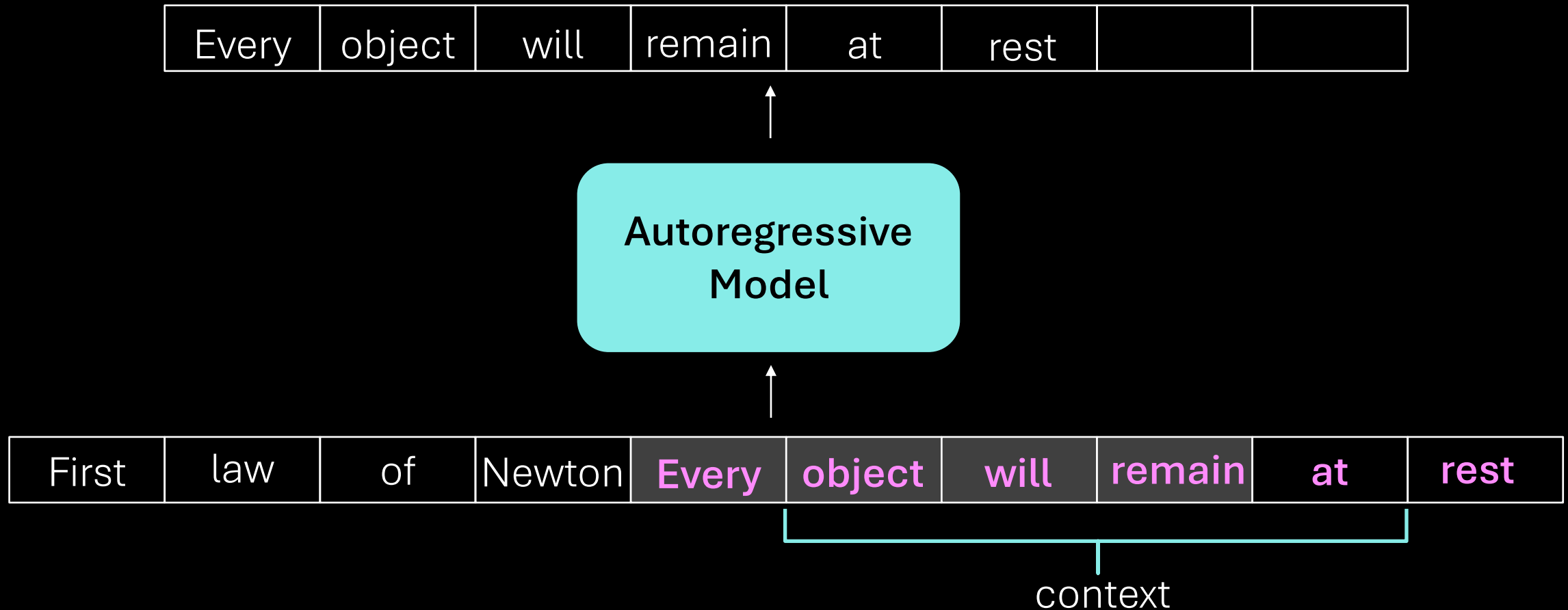
Autoregressive Model

Unidirectional processing and a fixed-length sliding window are characteristic of a general autoregressive model.



Autoregressive Model

Unidirectional processing and a fixed-length sliding window are characteristic of a general autoregressive model.



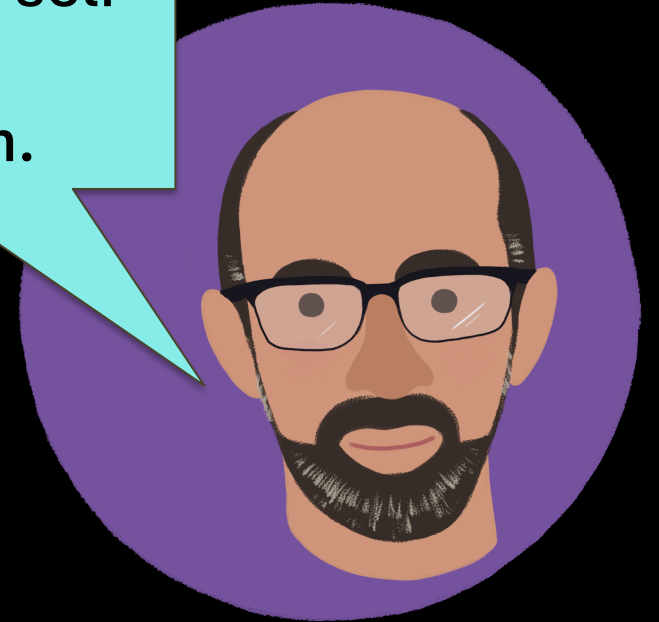
GPT | Training

- › The goal when training an autoregressive model is to prevent the model from getting a **sneak peek** at **future** tokens during training.



But how can we do this if we don't train sequentially?

We modify the self-attention calculation.



Attention | Recap

But doesn't this mean that the model is looking at the future here if it is calculating the similarity score between the **first** and the **third** transformed token?



$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

Exactly! How about we mask the problematic scores?



Masked Self-Attention



This has been simplified for demonstration purposes.
(The softmax would also need to be modified).



$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

 $\odot$

Mask M

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

 $=$

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

Masked Self-Attention

So, now we have the following:

$$z_1 = \alpha_{11}V_1$$

$$z_2 = \alpha_{21}V_1 + \alpha_{22}V_2$$

$$z_3 = \alpha_{31}V_1 + \alpha_{32}V_2 + \alpha_{33}V_3$$

$$z_4 = \alpha_{41}V_1 + \alpha_{42}V_2 + \alpha_{43}V_3 + \alpha_{44}V_4$$

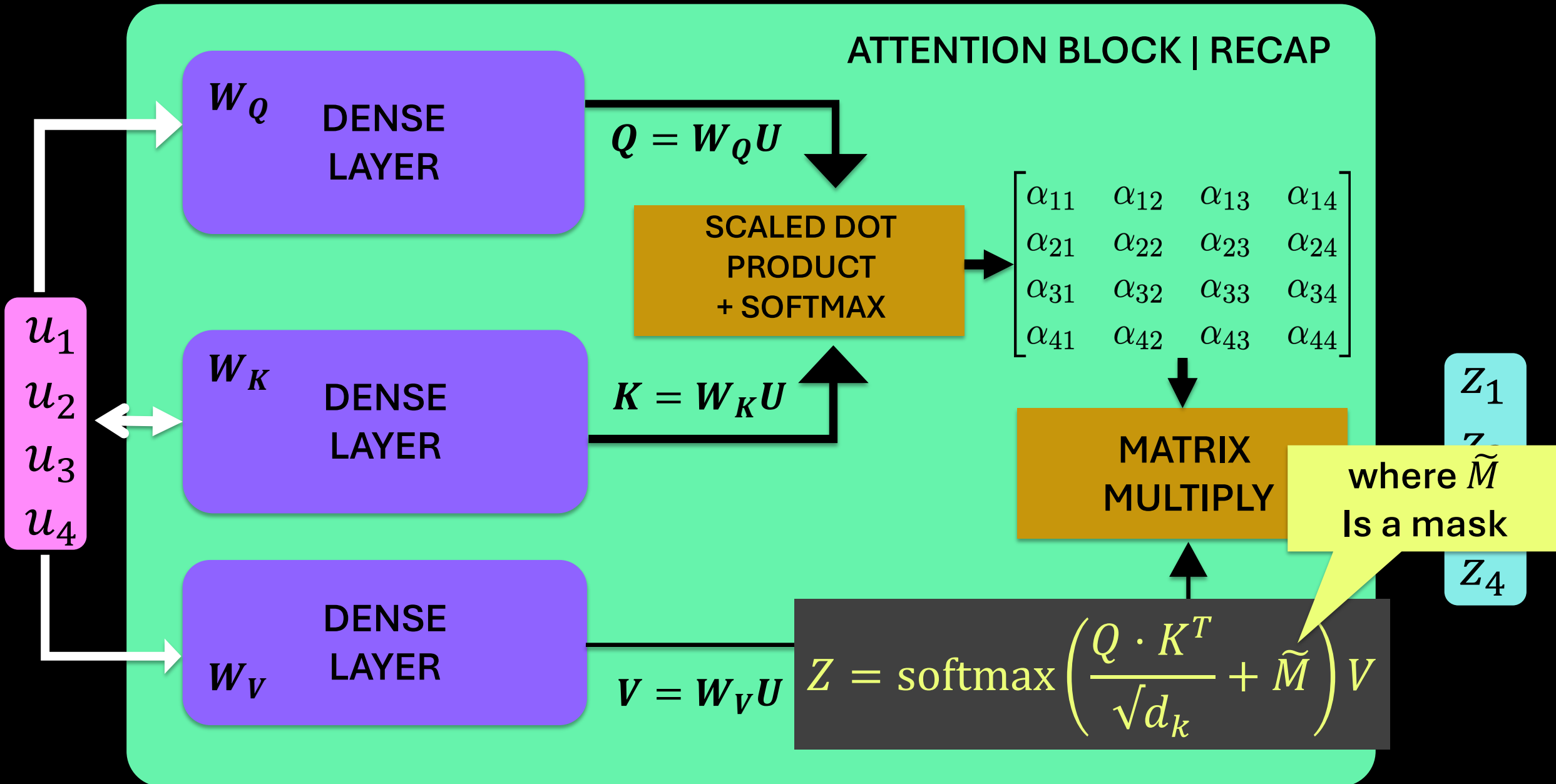
Yes, let's look at the entire picture.



Woah! This indeed is magic.
Now, no token has information
from the future tokens.



ATTENTION BLOCK | RECAP



Masking | Two Views, One Result

Intuitive View

Multiply the attention weights by M
(1s on and below the diagonal, 0s
above)

$$\alpha \odot M$$

1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

1 = keep, 0 = drop (after softmax)

Real Implementation

Add $\tilde{M}$ to the scores before softmax
(0s on and below the diagonal, $-\infty$
above)

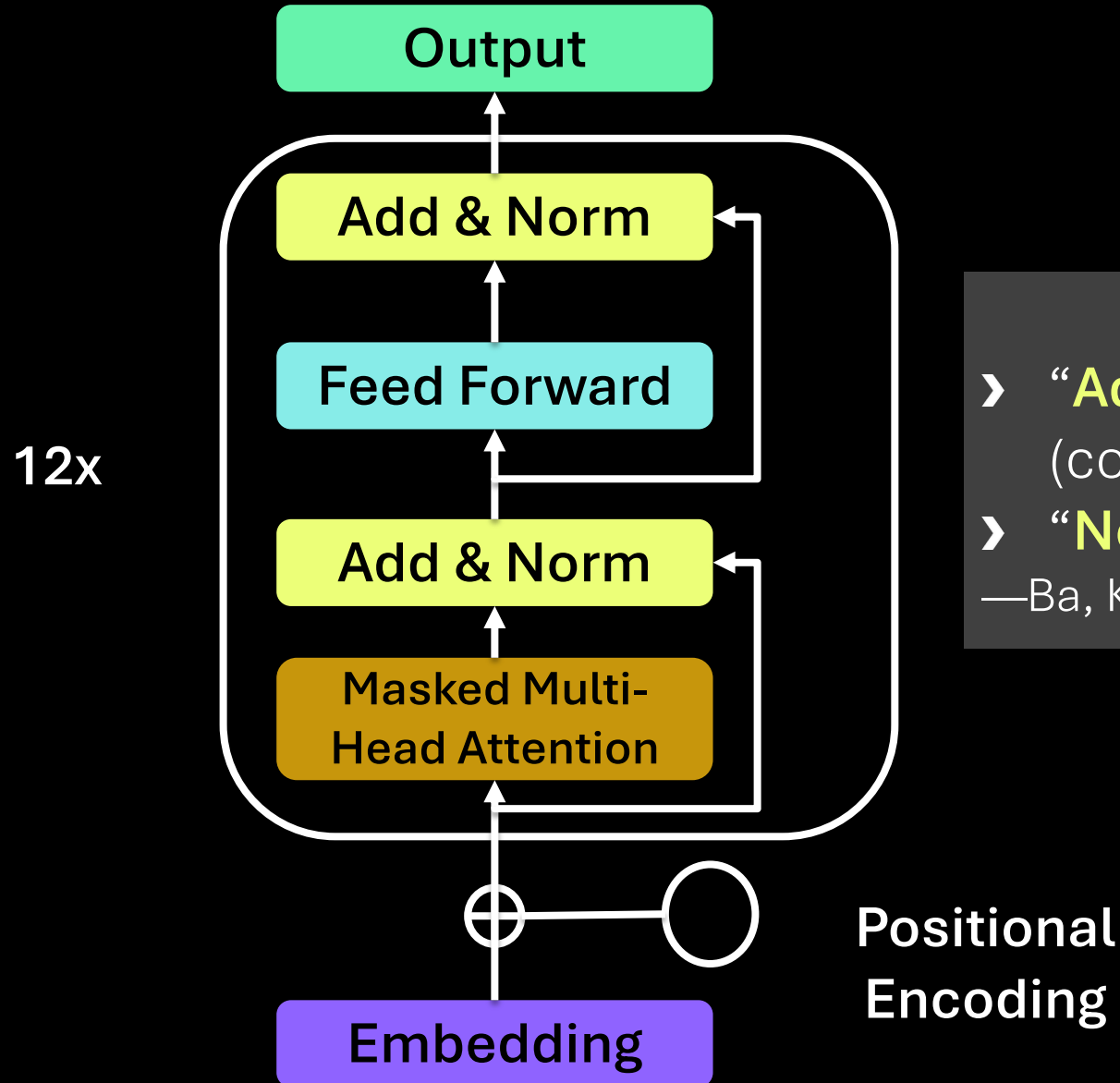
$$\text{softmax}(\text{scores} + \tilde{M})$$

0	$-\infty$	$-\infty$	$-\infty$
0	0	$-\infty$	$-\infty$
0	0	0	$-\infty$
0	0	0	0

$\text{softmax}(-\infty) = 0 \rightarrow \text{future dropped}$

Either way, future positions get zero weight — the same lower-triangular result.
No token can see the future.

GPT | The Complete Picture (GPT-1)



† Add & Norm:

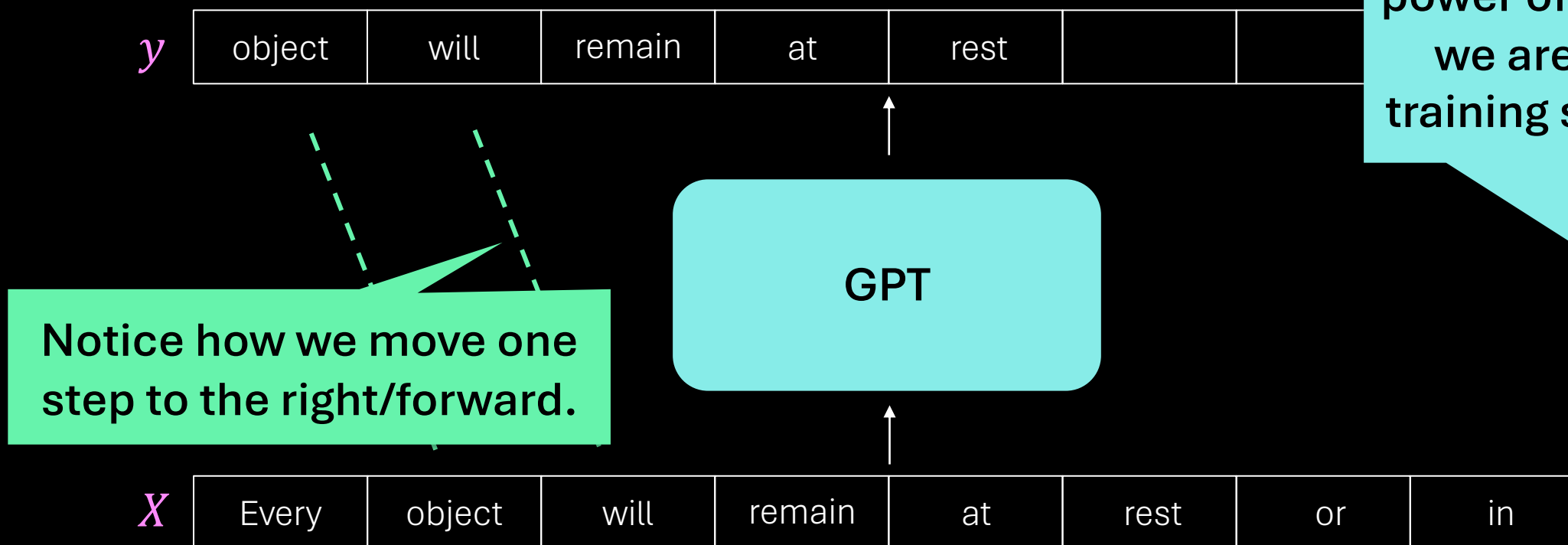
- “**Add**” is the Residual Connection (covered with BERT).
 - “**Norm**” is Layer Normalization.
- Ba, Kiros & Hinton, 2016 (arXiv:1607.06450).

GPT | Training

With the mask in place, GPT trains as a plain next-word predictor.

Input (X) = the sentence.

Target (y) = the same sentence, shifted one word to the left.

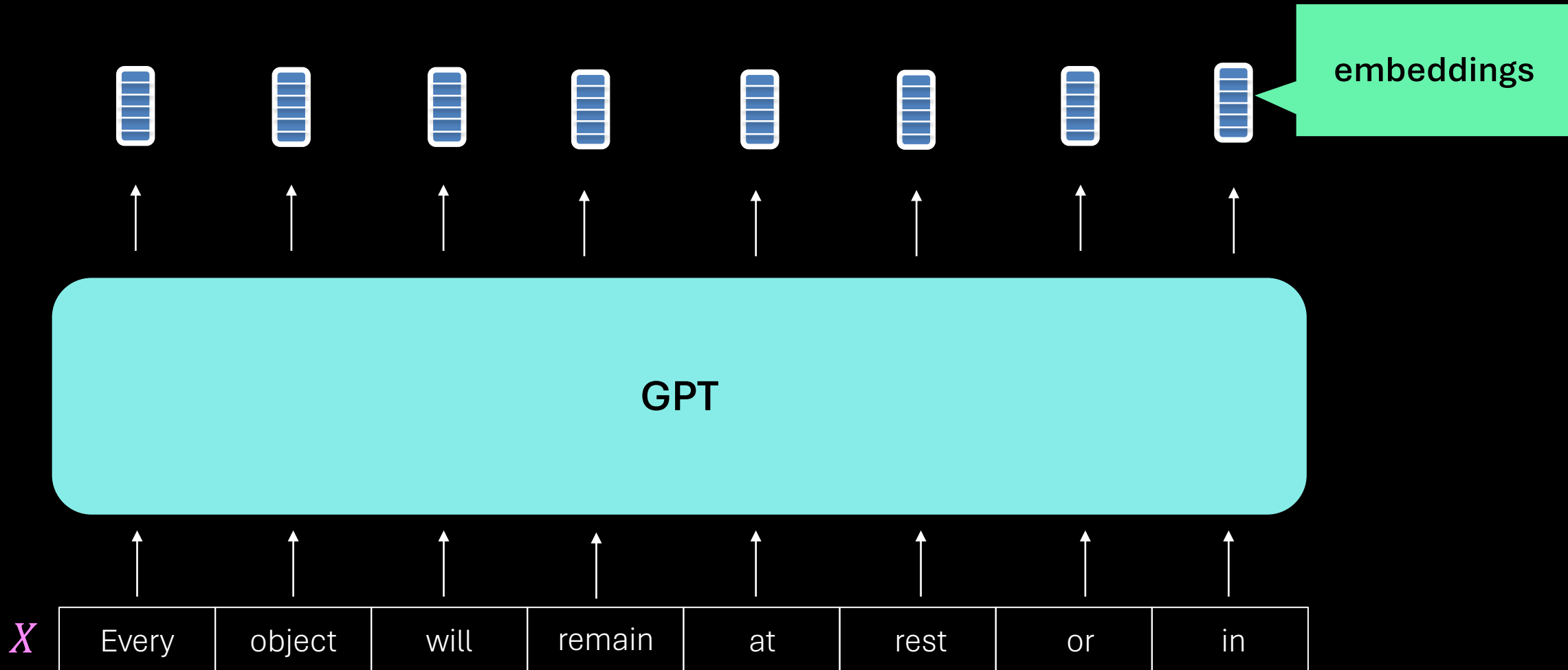


Now we can even make use of the power of GPUs, since we are no longer training sequentially.



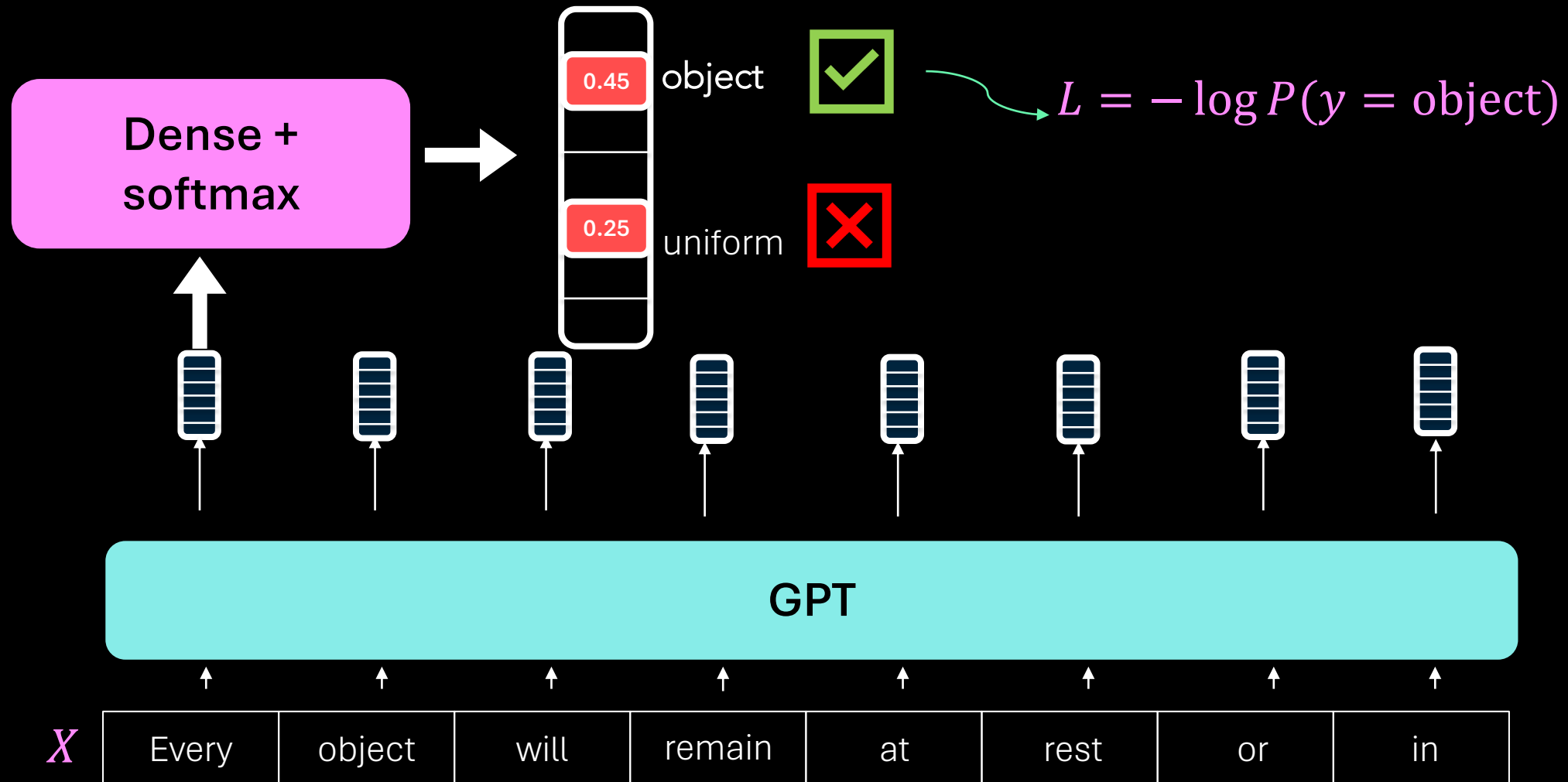
GPT | Training

Let us now look at what the training step looks like.



GPT | Training

Similarly, we calculate the loss for every output embedding.

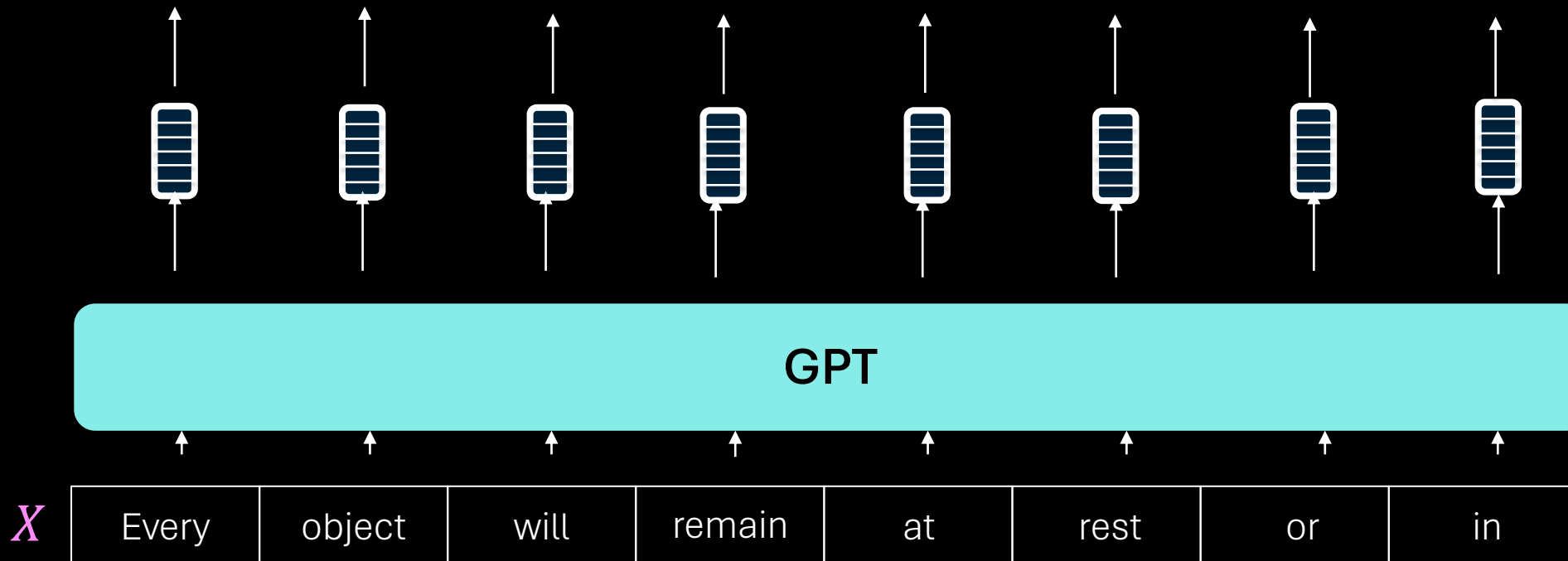


GPT | Training

Similarly, we calculate the loss for every output embedding.

object	will	remain	at	rest	or	in
--------	------	--------	----	------	----	----

$$L = -\log P(y = \text{object}) \quad L = -\log P(y = \text{remain}) \quad \dots \quad \dots \quad \dots \quad L = -\log P(y = \text{uniform})$$



GPT | Inference

There are two ways GPT can generate text:

1. **Conditional**: We provide a starting prompt (context) and the model continues generating the text from there.

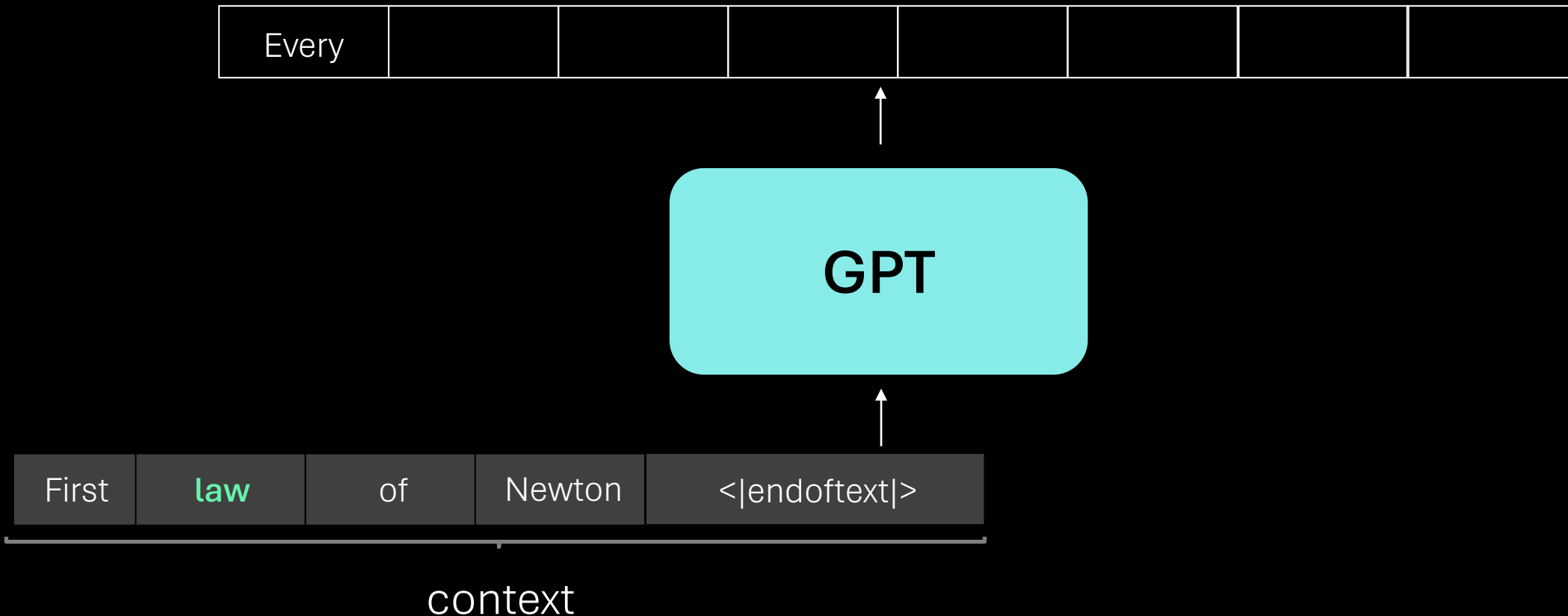


This is a special token used by the model to mark the end of the input or the starting point for predictions.

GPT | Inference

There are two ways GPT can generate text:

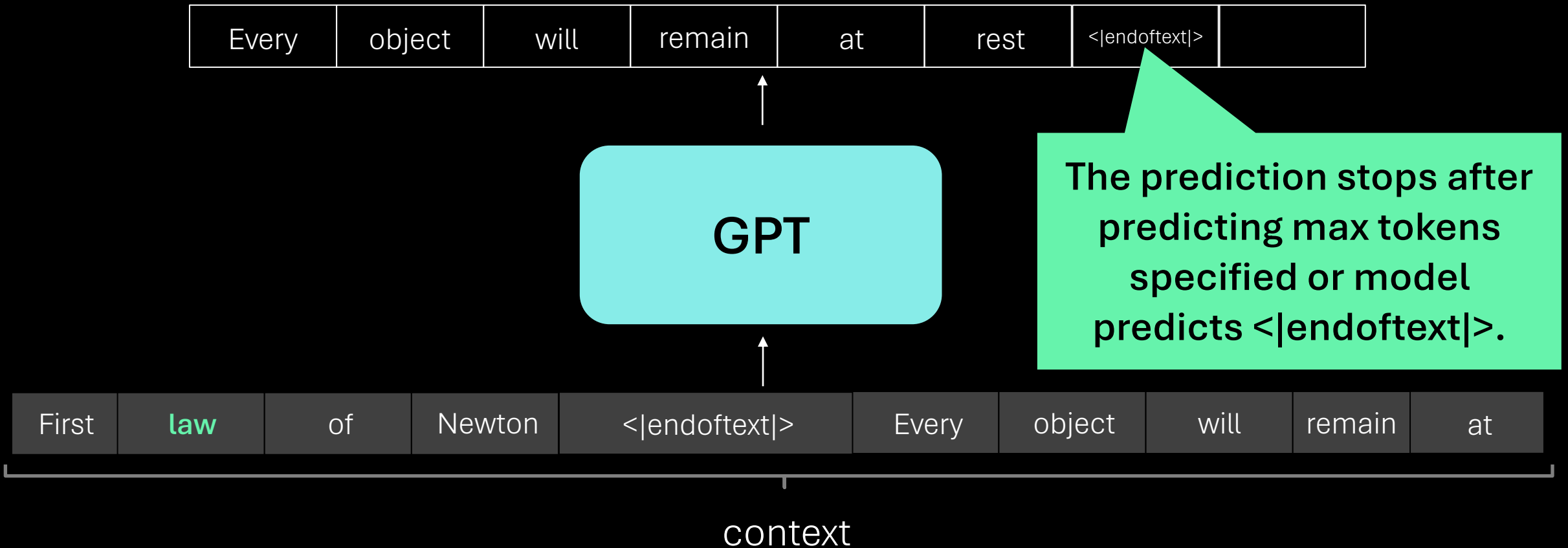
1. **Conditional**: We provide a starting prompt (context) and the model continues generating the text from there.



GPT | Inference

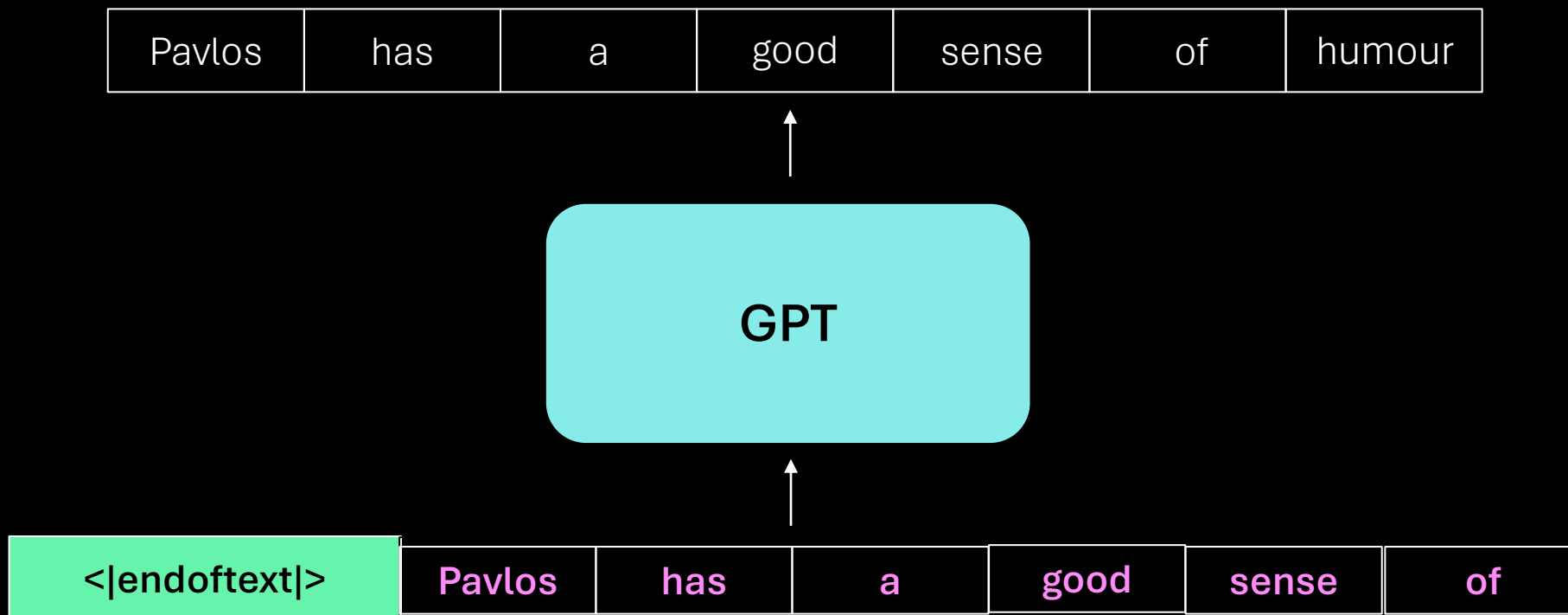
There are two ways GPT can generate text:

1. **Conditional**: We provide a starting prompt (context) and the model continues generating the text from there.

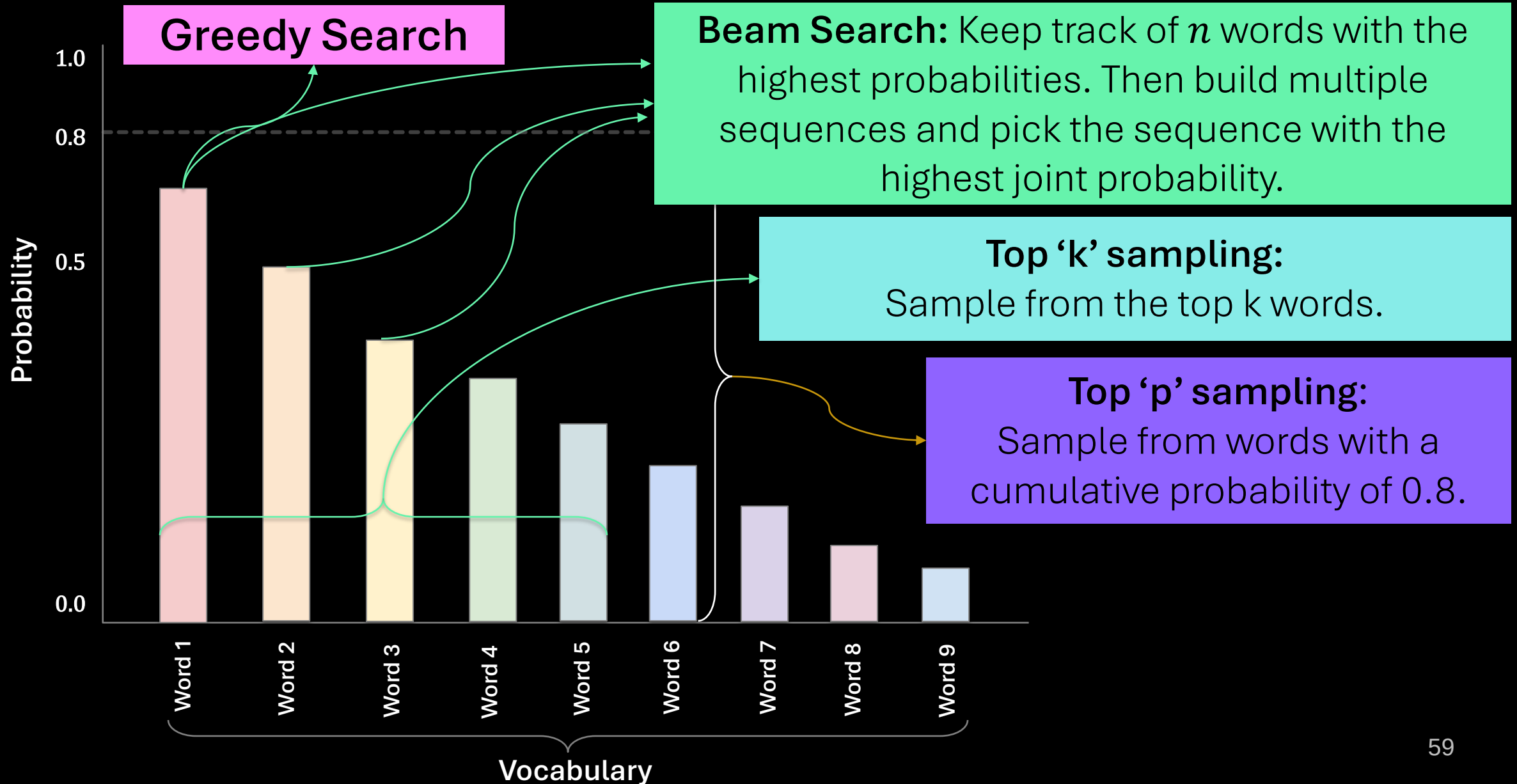


GPT | Inference

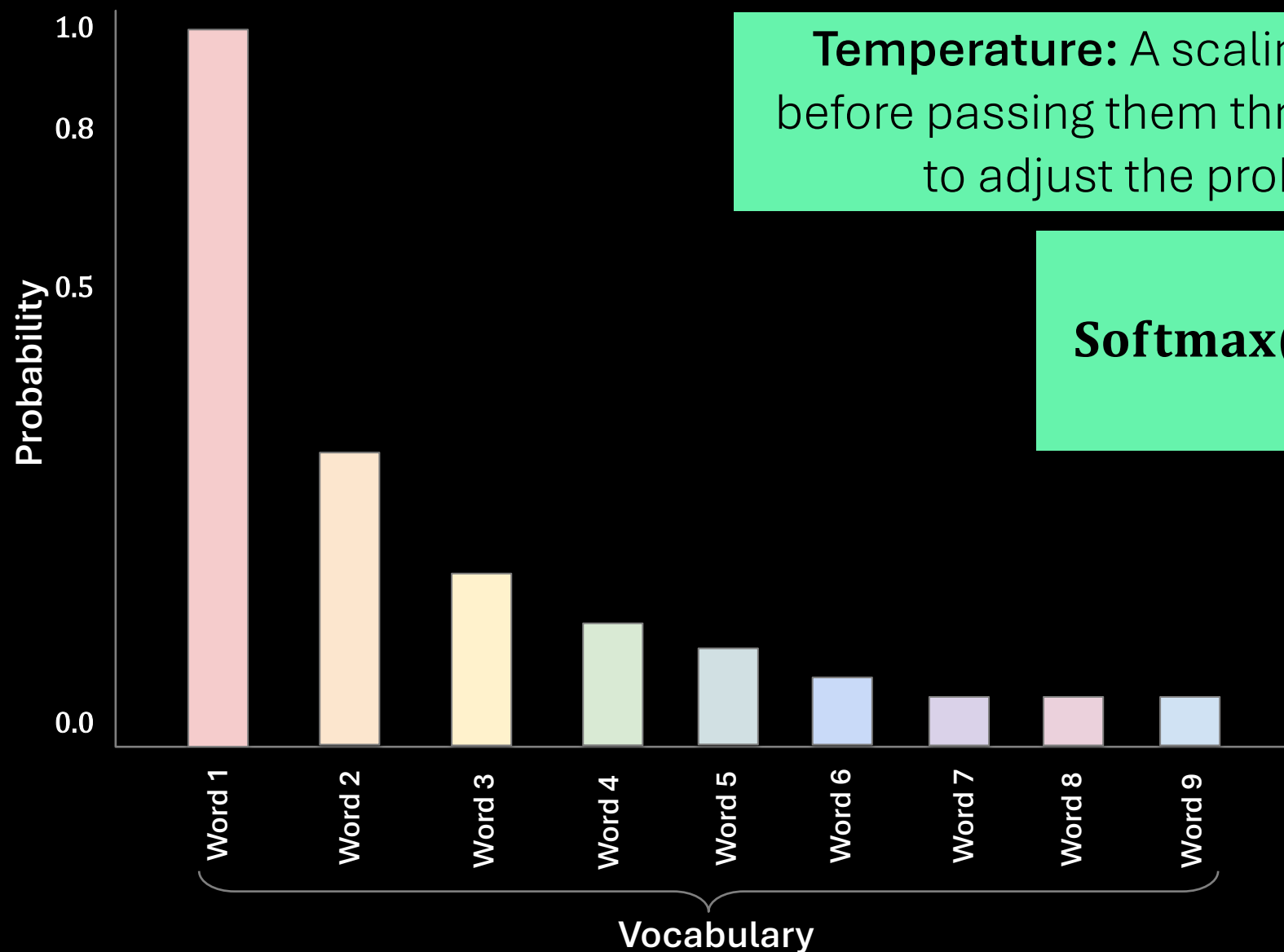
2. **Unconditional:** We start with `<|endoftext|>`, which acts as a special start token to have the model generate words on a topic of its own choosing.



GPT | Inference



GPT | Inference



Temperature: A scaling factor applied to logits before passing them through the softmax function to adjust the probability distribution.

$$\text{Softmax}(\mathbf{z}_i; \mathbf{T}) = \frac{\exp\left(-\frac{\mathbf{z}_i}{T}\right)}{\sum_{j=1}^n \exp\left(-\frac{\mathbf{z}_j}{T}\right)}$$

Low Temperature:

- Sharpens probability distribution.
- Reduces randomness in predictions.

Evolution of GPT

Since its inception in 2018, GPT has received some major upgrades in terms of performance, thanks to the increasing number of parameters with each iteration.

Model	Parameters	Layers	Hidden Dim / Width	Context / Seq Length	Batch Size*	Effective Tokens per Batch
GPT-1	~117M	12	768	512	—	
GPT-2 (1.5B)	1.5B	48	1600	1024	512 sequences	$\sim 512 \times 1024 =$ $\sim 0.5\text{M tokens}$
GPT-3 (175B)	175B	96	12,288	2048	3.2M tokens	$\sim 3.2\text{M tokens}$

GPT | Key Takeaways

- › **Autoregressive language model** — predicts the next token using causal, left-to-right attention.
- › **Masked self-attention** — the key trick: hide future tokens, so we can train on the whole sequence in parallel.
- › **Training** — predict the next token with a one-shifted target; cross-entropy loss; GPU-friendly.
- › **Generation** — autoregressive at inference, one token at a time; decoding knobs: greedy, beam, top-k / top-p, temperature.
- › **Scales** — 117 M (GPT-1) → 1.5 B (GPT-2) → 175 B+ (GPT-3); bigger models and data unlock emergent skills.

Up Next | From GPT to ChatGPT



A raw GPT just continues text.
Next, we turn it into an
assistant that follows instructions
— ChatGPT.

- › Instruction tuning and RLHF (learning from human feedback)
- › System prompts that steer the model
- › Multi-turn conversation and memory