LARGE LANGUAGE MODELS

BERT and GPT

Presented By Ahmed Baari

How Traditional NLP Methods Failed

- Classical NLP, Markov Models, and Statistical Methods were groundbreaking but struggled with deeper language understanding.

- Neural Networks and deep learning addressed some gaps but introduced new challenges.

Classical Probability Theory - Failed

Elementary Probability Theory: Relied heavily on rigid assumptions.

Problems:

- Couldn’t capture context beyond adjacent words.
- Limited in handling ambiguous meanings or unseen phrases.

Impact: Difficulty in real-world applications like Machine Translation or Speech Recognition.

Information Theory - Failed

- **Essential Information Theory:** Helped quantify language patterns.

Problems:

- Failed to model long-term dependencies.
- Struggled with contextual understanding over large text bodies.

Result: Loss of meaning in complex sentences and contexts.

Word Sense Disambiguation – Failed

- **Supervised/Dictionary-Based Methods:** Attempted to resolve word ambiguity.

Problems:

- Required massive labeled data.
- Poor generalization to new or unseen data.

Consequence: Struggled with disambiguation in dynamic, evolving text like social media or news.

Markov Models – Failed

- **Hidden Markov Models (HMMs):** Useful in modeling sequences.

Problems:

- Could only capture immediate prior context.
- Failed with long-range dependencies in language (e.g., remembering subject-verb agreement across clauses).

Result: Failed performance in translation and speech recognition beyond simple tasks.

PCFG - Failed

- **Probabilistic Context-Free Grammars:** Extended CFGs with probabilities.

Problems:

- Limited flexibility in handling natural language variation.
- Performance degraded in ambiguous or ungrammatical inputs.

Impact: Inefficiency in real-world parsing of complex sentences.

Neural Networks – Early Success

Feedforward Neural Networks: Used for classification and language modeling.

Problems:

- Failed to capture sequential nature of language.
- XOR problem showed limitations of simple perceptrons for complex patterns.

Impact: Could not effectively handle dependencies across time (sentences).

RNNs and LSTMs- Almost There!

Recurrent Neural Networks (RNNs): Designed to handle sequences.

Problems:

- Vanishing Gradient Problem: Information from earlier inputs would be lost in long sequences.
- Limited in modeling long-range dependencies.

Consequence: Struggled with large-scale tasks like Machine Translation, sentiment analysis, or text generation.

Transformers to the Rescue!

Solved vanishing gradients with self-attention.

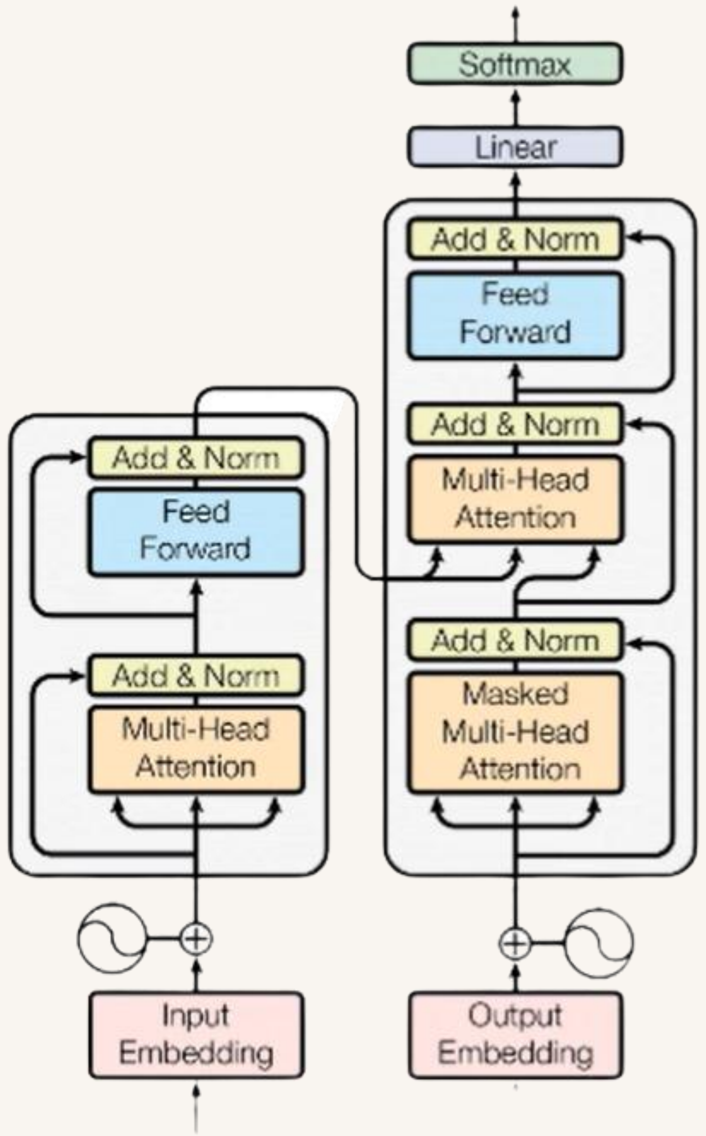
- Allowed parallelization (better scalability).
- No sequential processing bottleneck.

- **Large Language Models:**
- Trained on vast data with billions of parameters (BERT, GPT).
- Finally captured long-range dependencies, nuanced context, and real-world language.

The Transformer Architecture

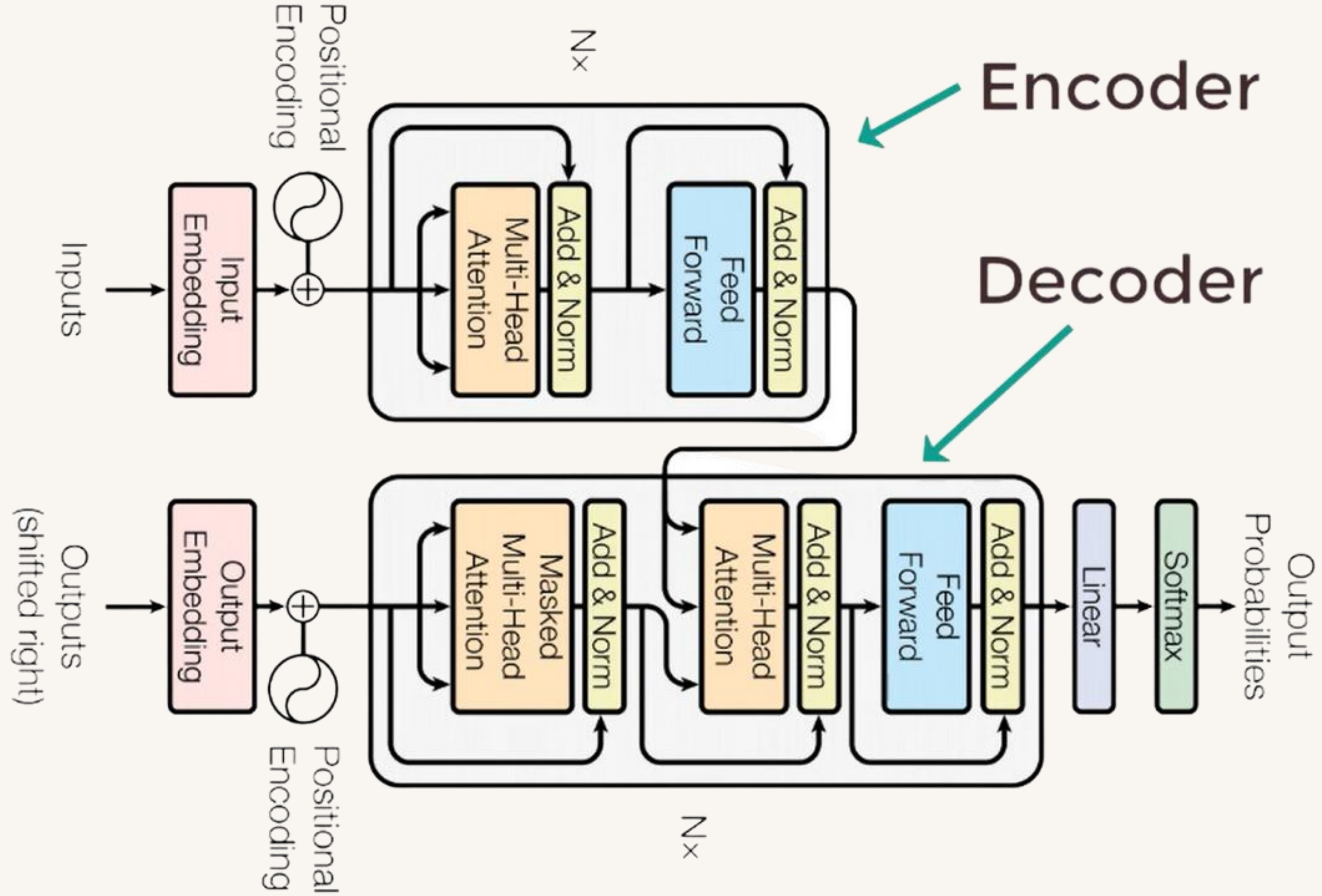
Trasformers can perform various tasks. The above example demonstrates the transformer architecture performing Translation.

The Transformer Architecture



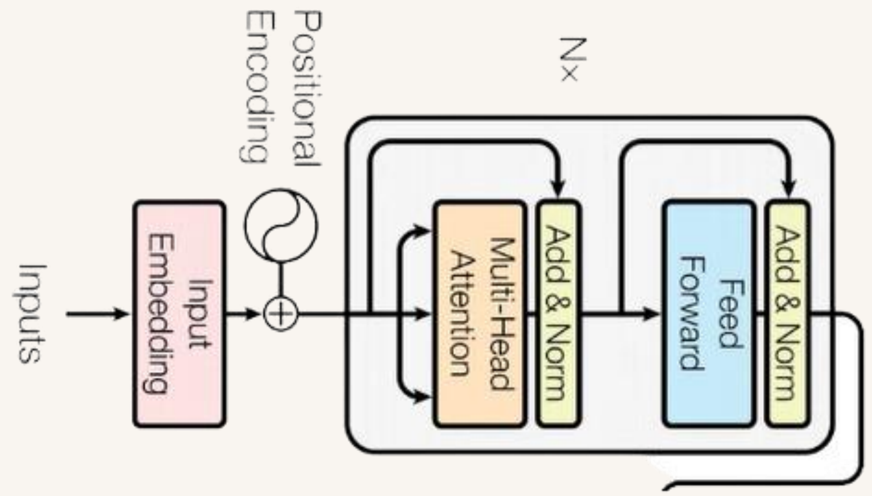
The Transformer Architecture

The transformer architecture contains 2 components: Encoder, Decoder



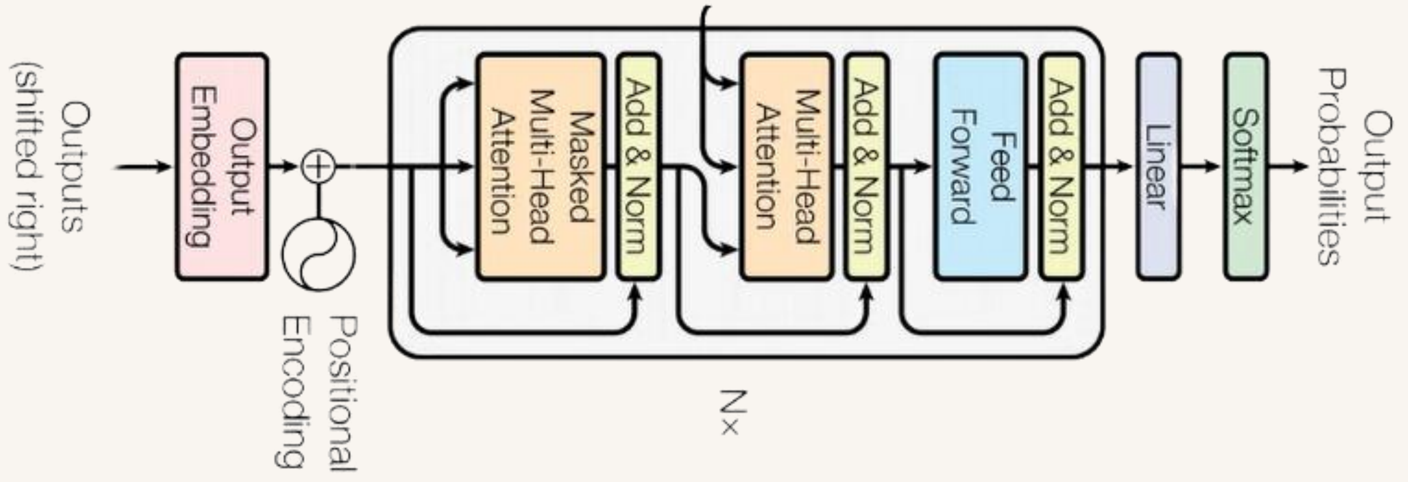
The Transformer Architecture

Transformer Flow



What is English? What is context?

What is language!

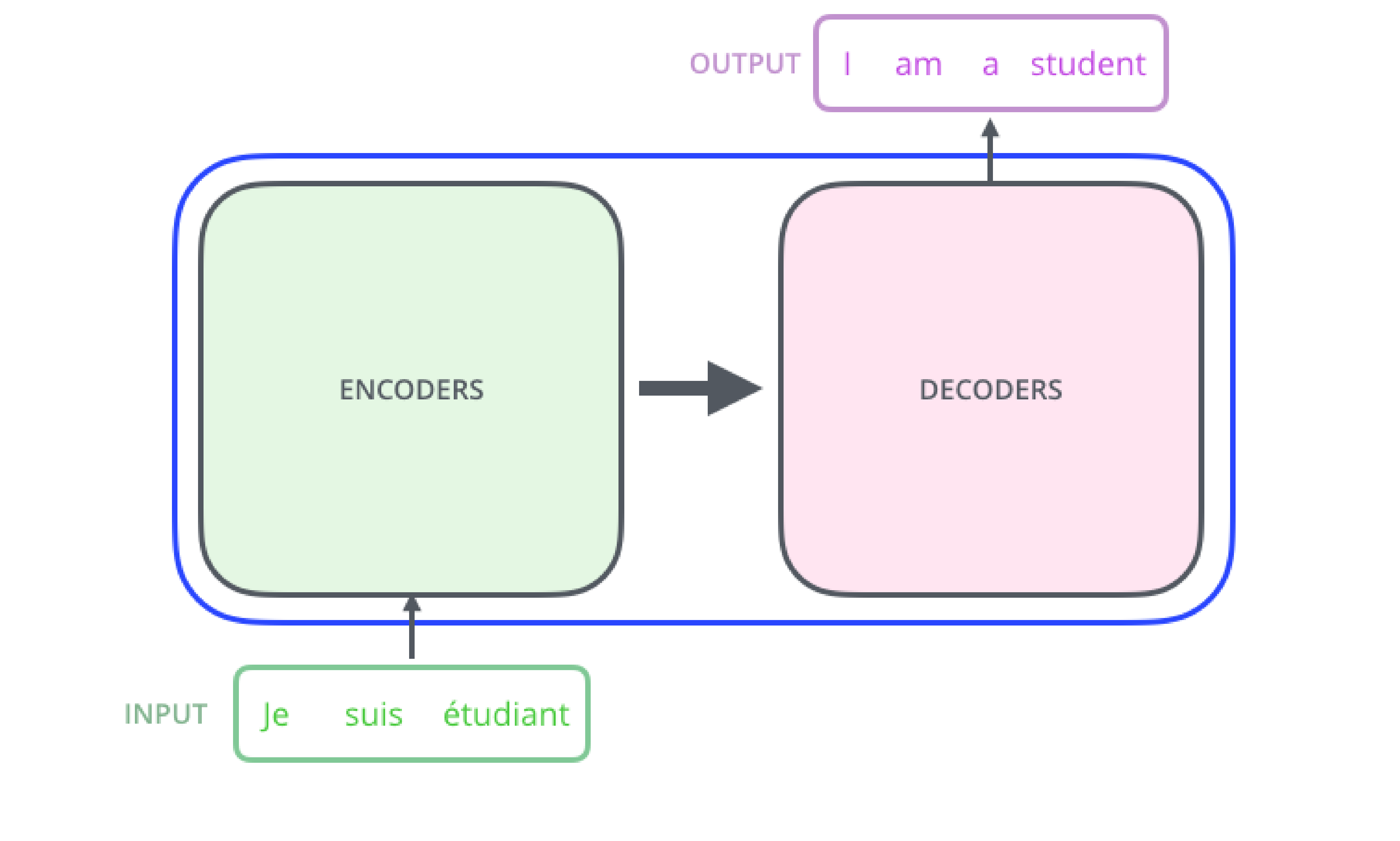


How to map English words to French words?

What is language!

The Transformer Architecture

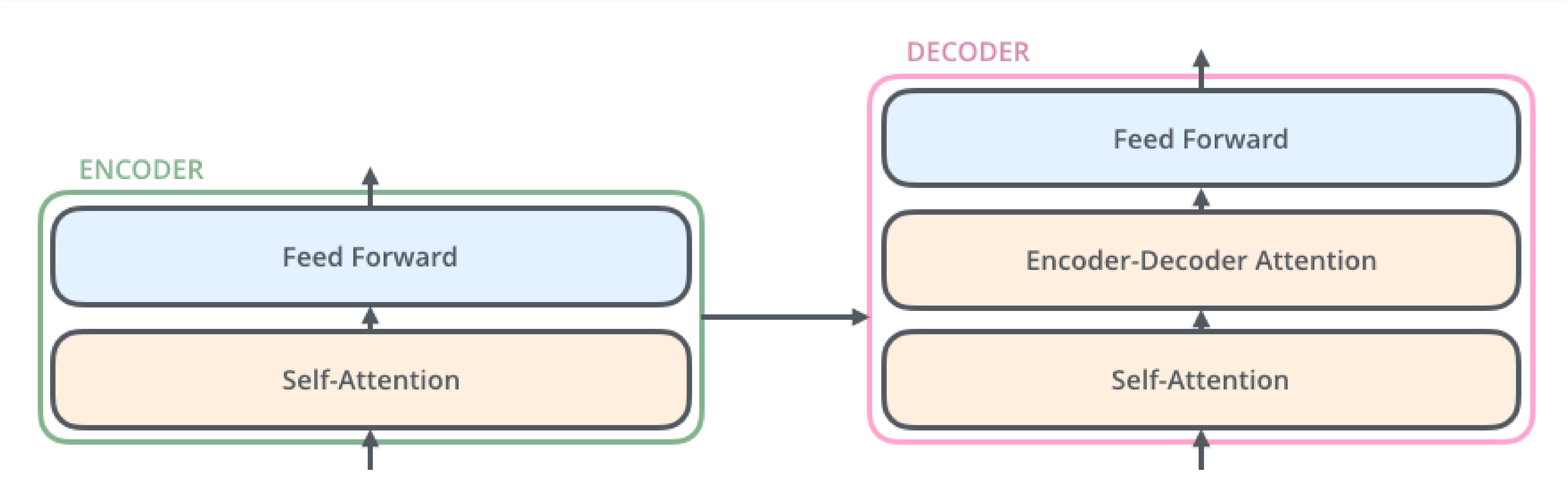
Let us simplify this diagram



Encoder and Decoder

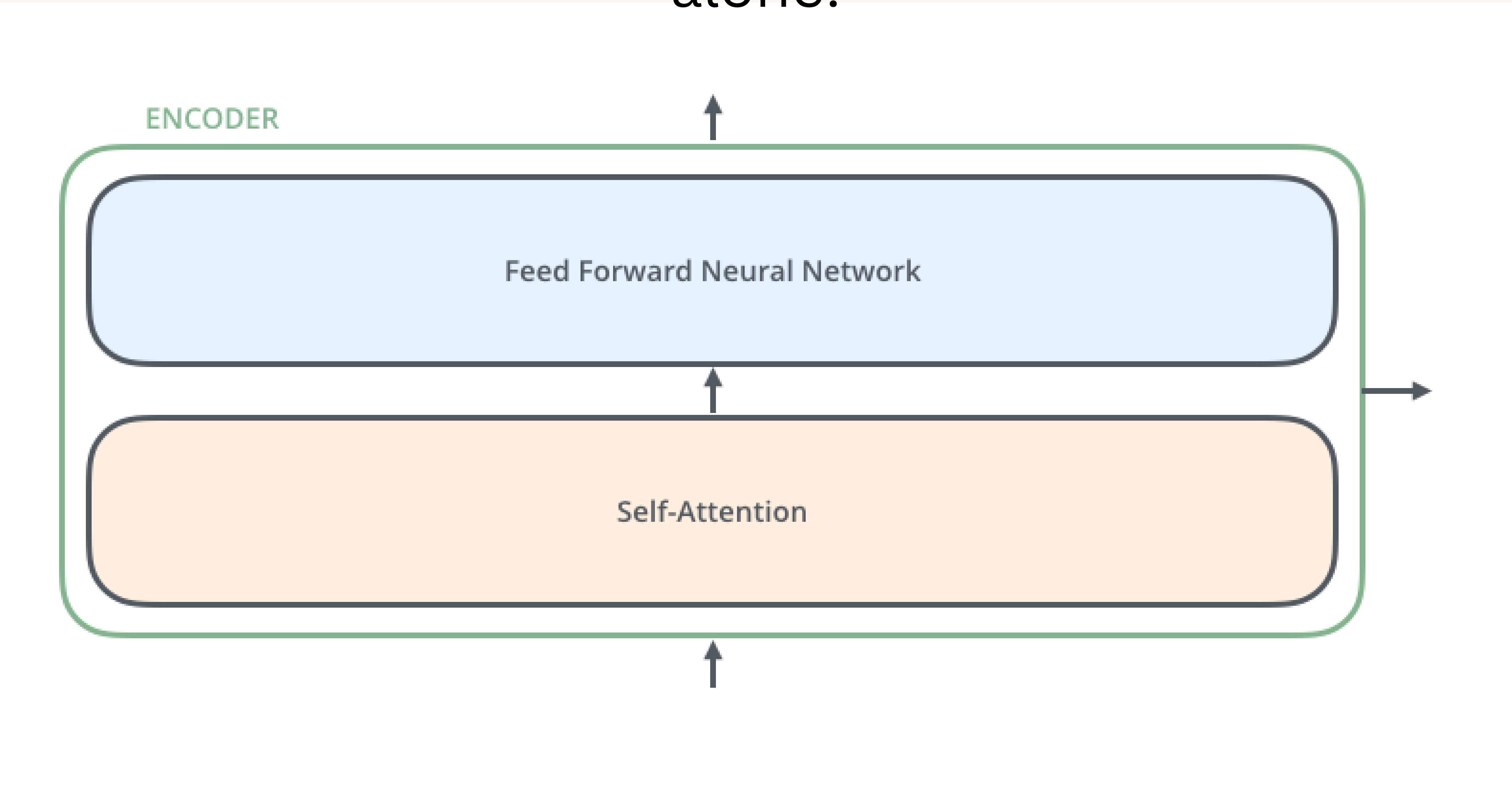
Both encoder and decoder contain a **self-attention** layer and a **feed forward** layer.

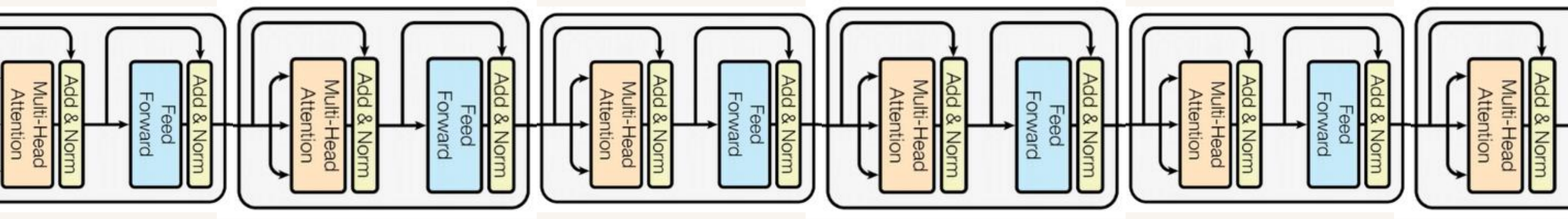
The decoder additionally contains a layer which works with both the encoder's attention values, and the decoder's attention values. This layer is called the **Encoder Decoder Attention** layer.



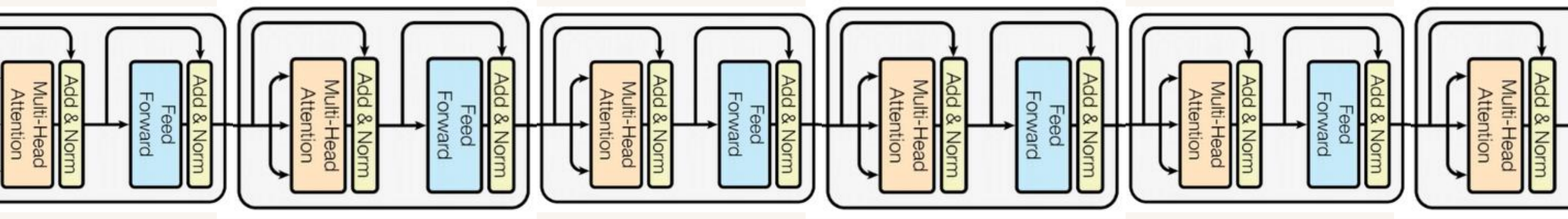
Encoder

First, let's focus on what we can do with this encoder alone!





What do we get when we stack many encoders, one above the other?



The B E R T Architecture!

Google
BERT

B
E
R
T

B E R T

Bidirectional

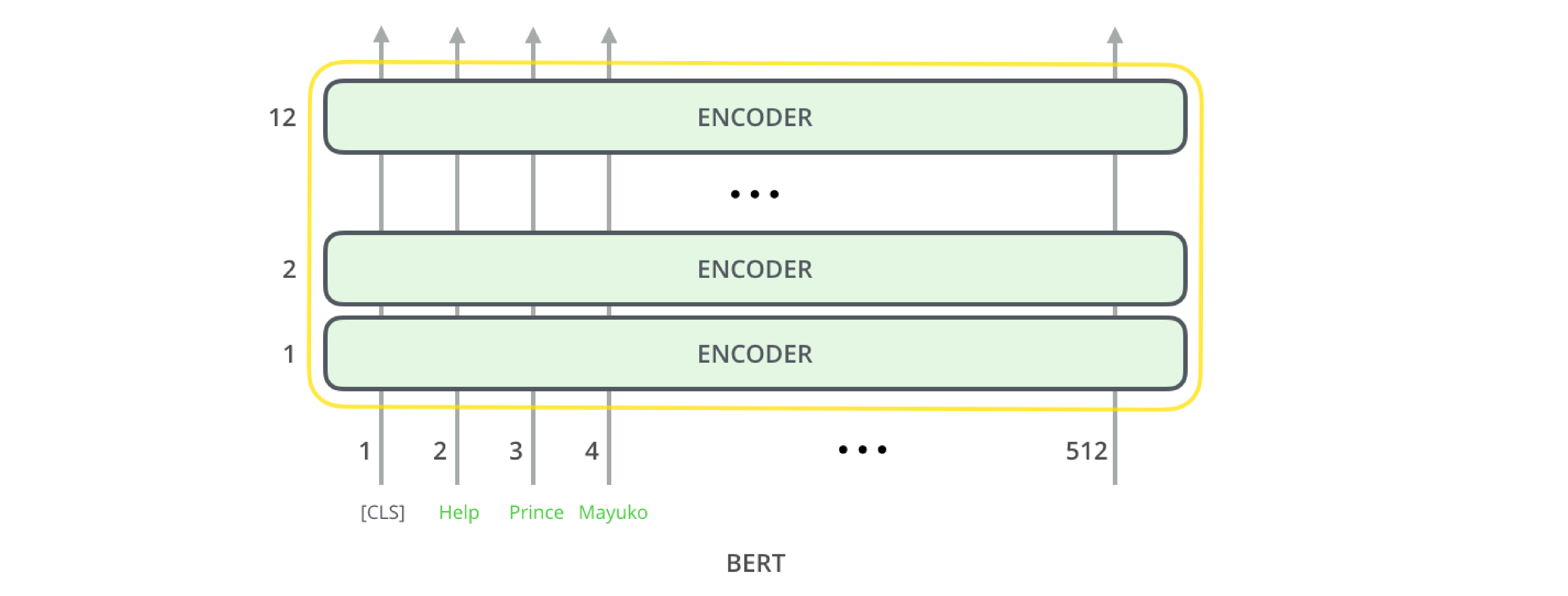
Encoder

Representations from

Transformers

What is BERT?

BERT (Bidirectional Encoder Representations from Transformers) is a powerful transformer-based language model that was designed to handle a wide range of natural language processing (NLP) tasks without needing task-specific architectures.



What can BERT do?

Problems to Solve

- Neural Machine Translation
- Question Answering
- Sentiment Analysis
- Text summarization

Needs Language understanding

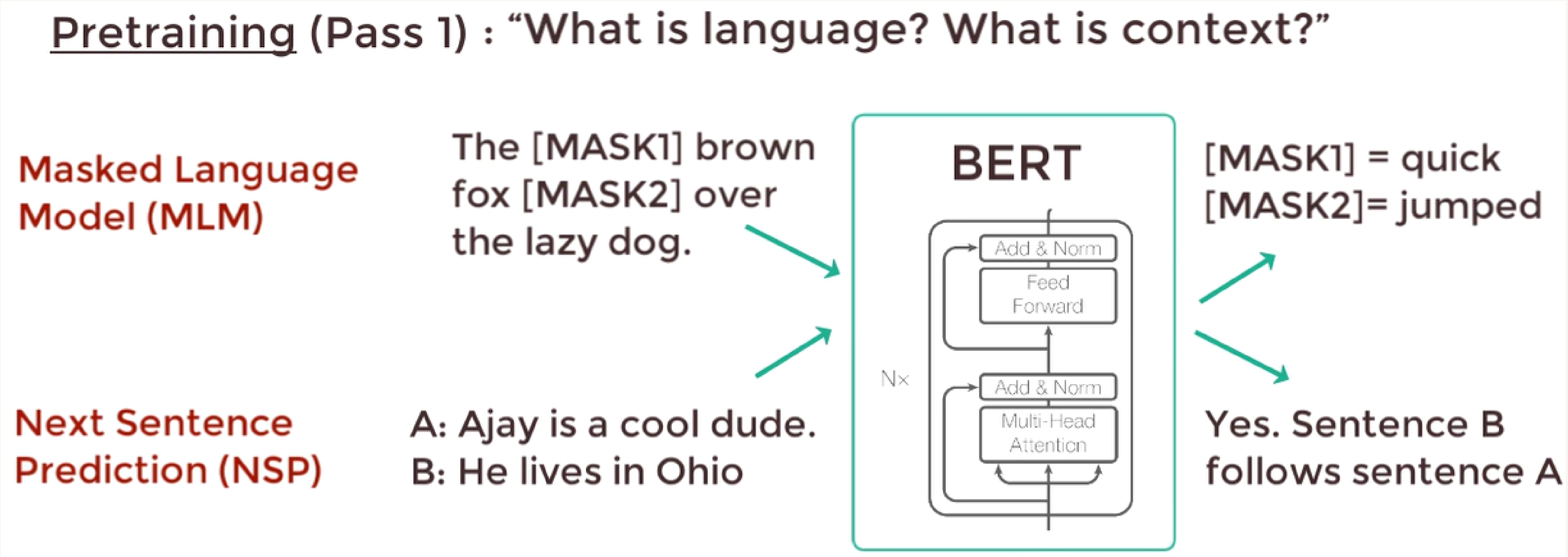
How to solve Problems (BERT Training)

- Pretrain BERT to understand langauge
- Fine tune BERT to learn specific task

1. Pre-Training BERT

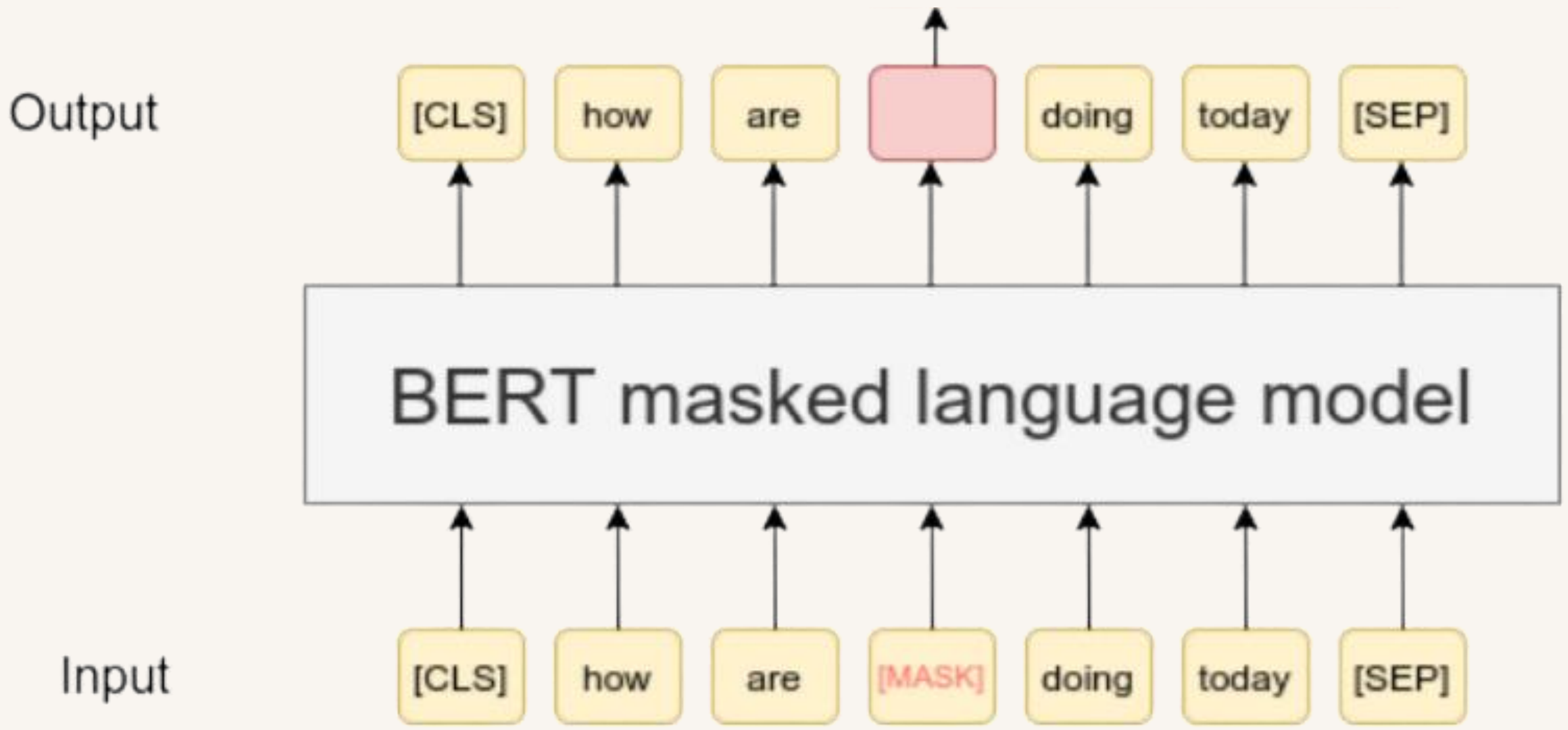
BERT is pre-trained on large datasets like Wikipedia and BookCorpus, using two main self-supervised learning tasks:

- Masked Language Modeling (MLM)
- Next Sentence Prediction (NSP)



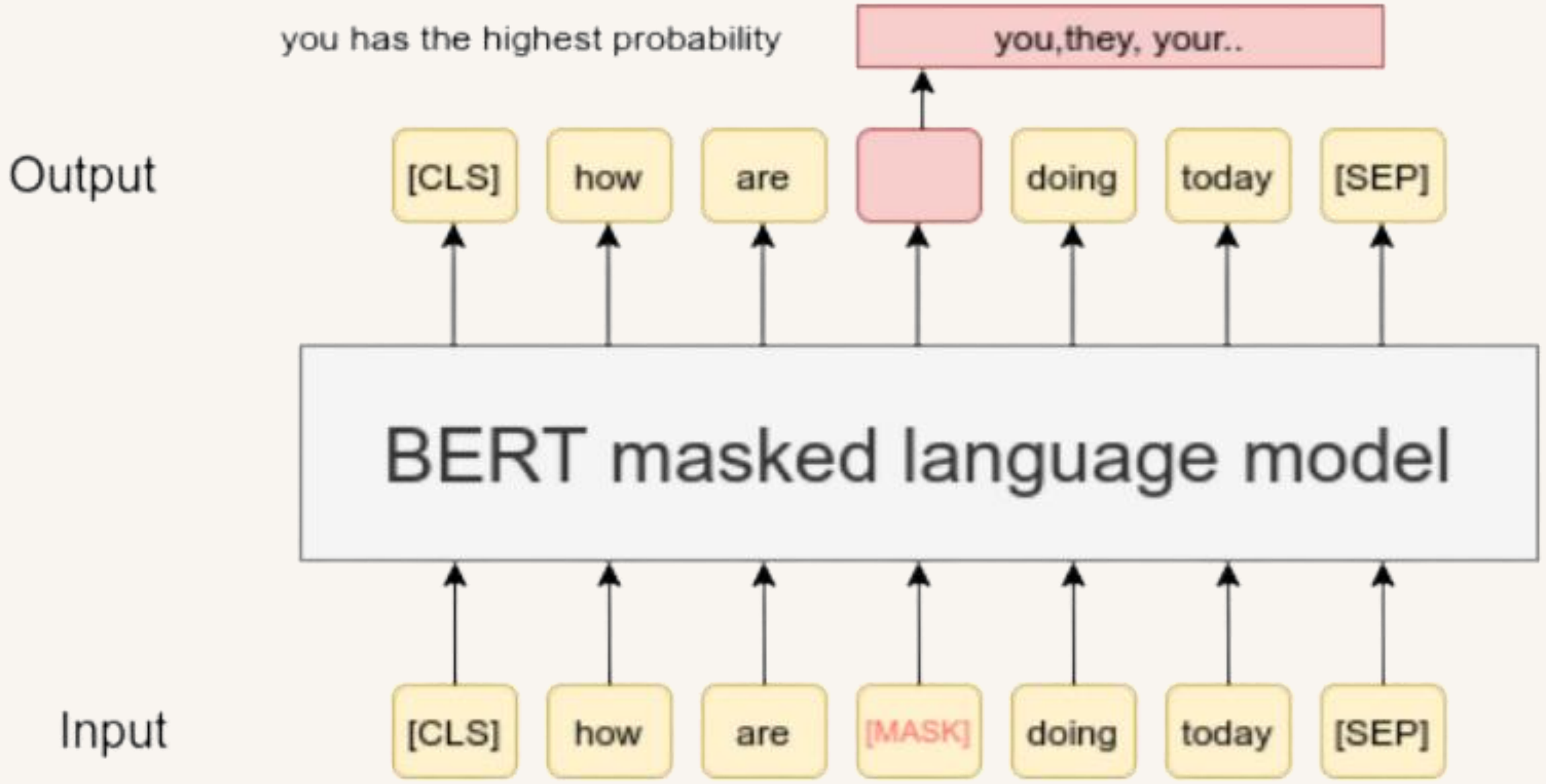
1A. Masked Language Modeling

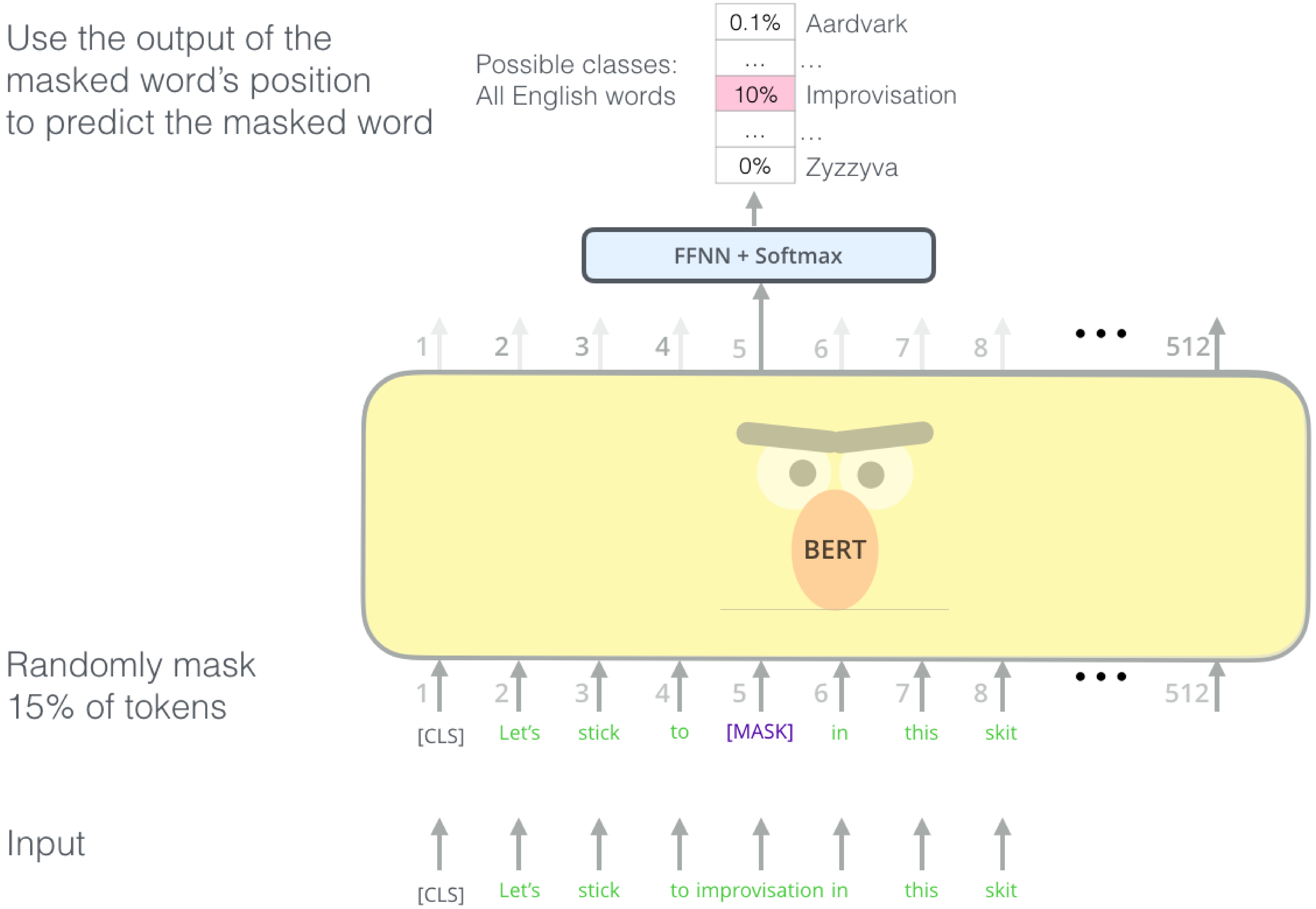
Random words in a sentence are replaced with a [MASK] token, and the model tries to predict these masked words. The idea is for BERT to learn context from both directions (left and right), unlike traditional models that read text in only one direction.



1A. Masked Language Modeling

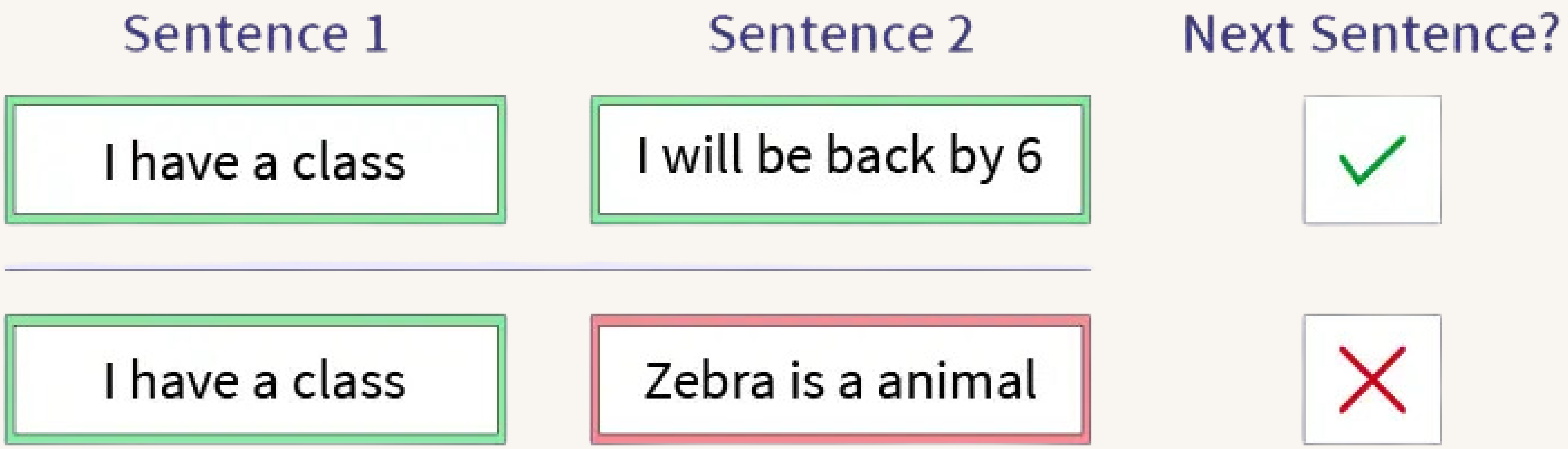
Random words in a sentence are replaced with a [MASK] token, and the model tries to predict these masked words. The idea is for BERT to learn context from both directions (left and right), unlike traditional models that read text in only one direction.




Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
1 2 3 4 5 6 7 8 ... 512
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

1B. Next Sentence Prediction

Binary Classification Task to predict if Sentence 2 comes after sentence 1.



1B. Next Sentence Prediction

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.

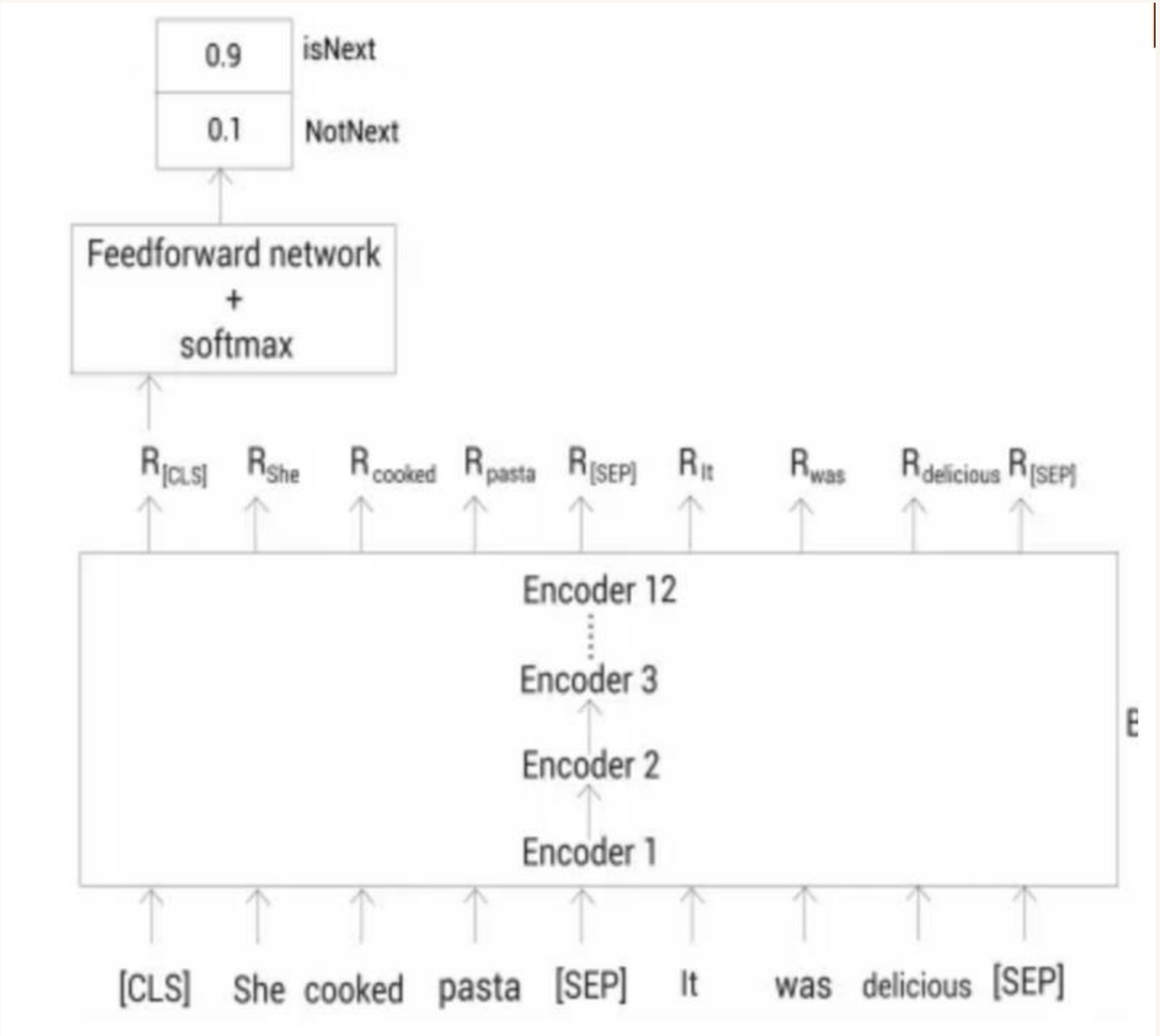
During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence.

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

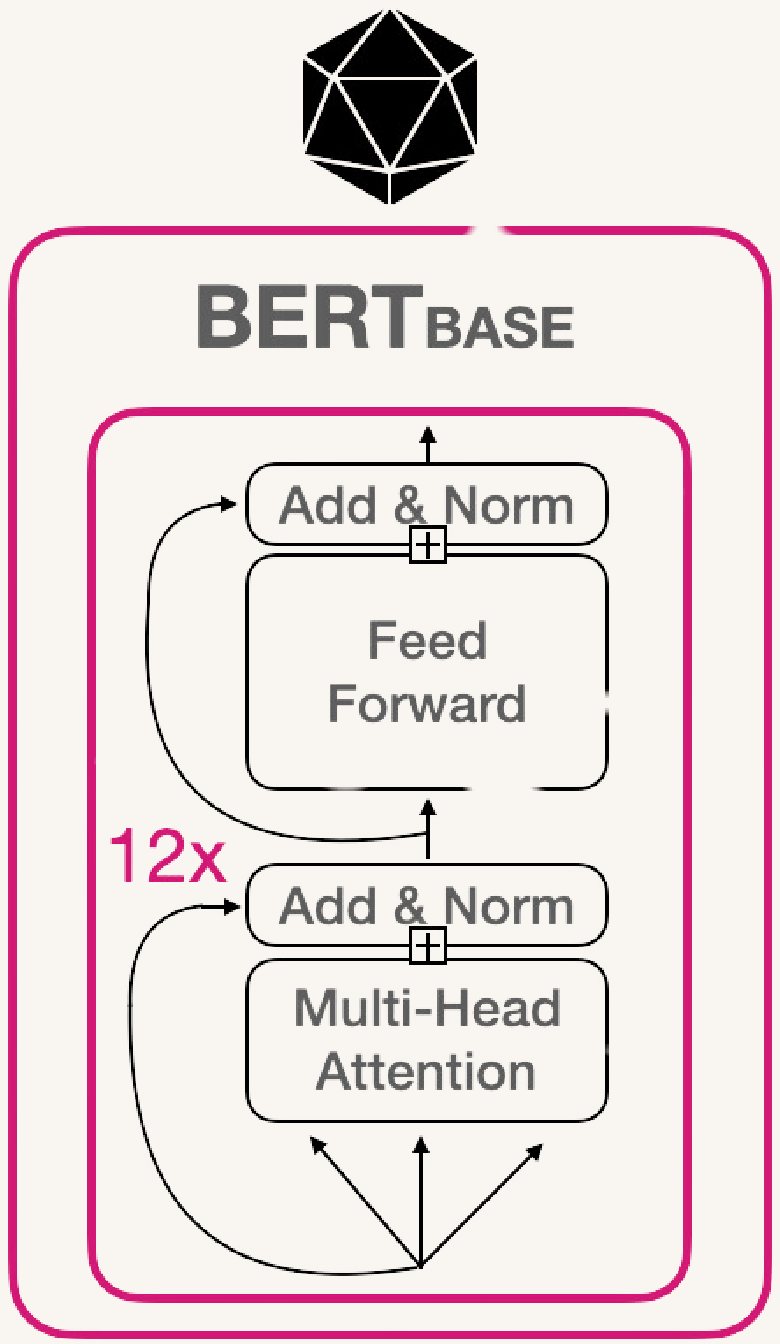
- We include a [CLS] and a [SEP] token.
- A sentence embedding indicating Sentence A or Sentence B is added to each token.
- A positional embedding is added to each token to indicate its position in the sequence.

NSP Simplified

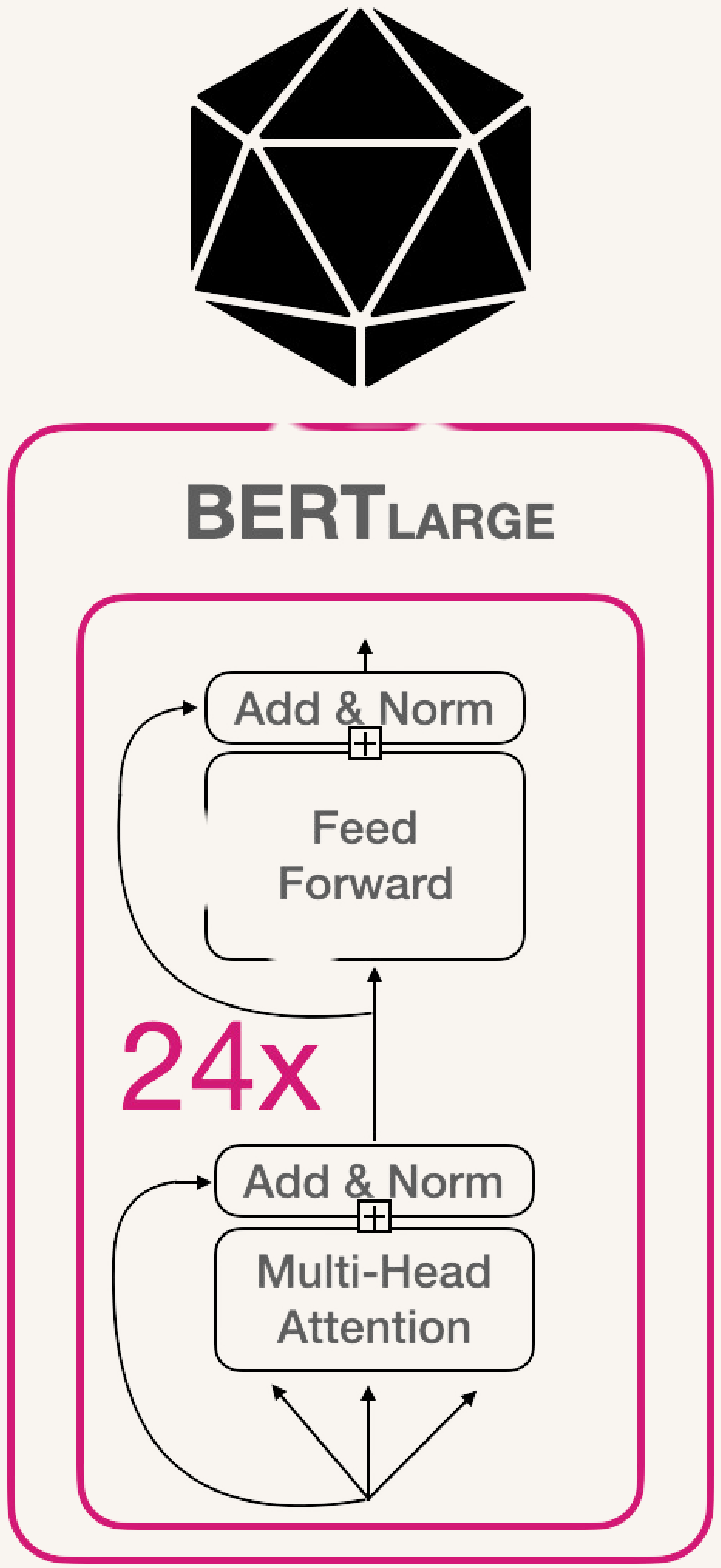
- We take the attention values for the CLS token and pass them through a softmax and a feed forward network.
- This is then used to predict wether the sentence that follows is the next sentence or not



BERT Size & Architecture

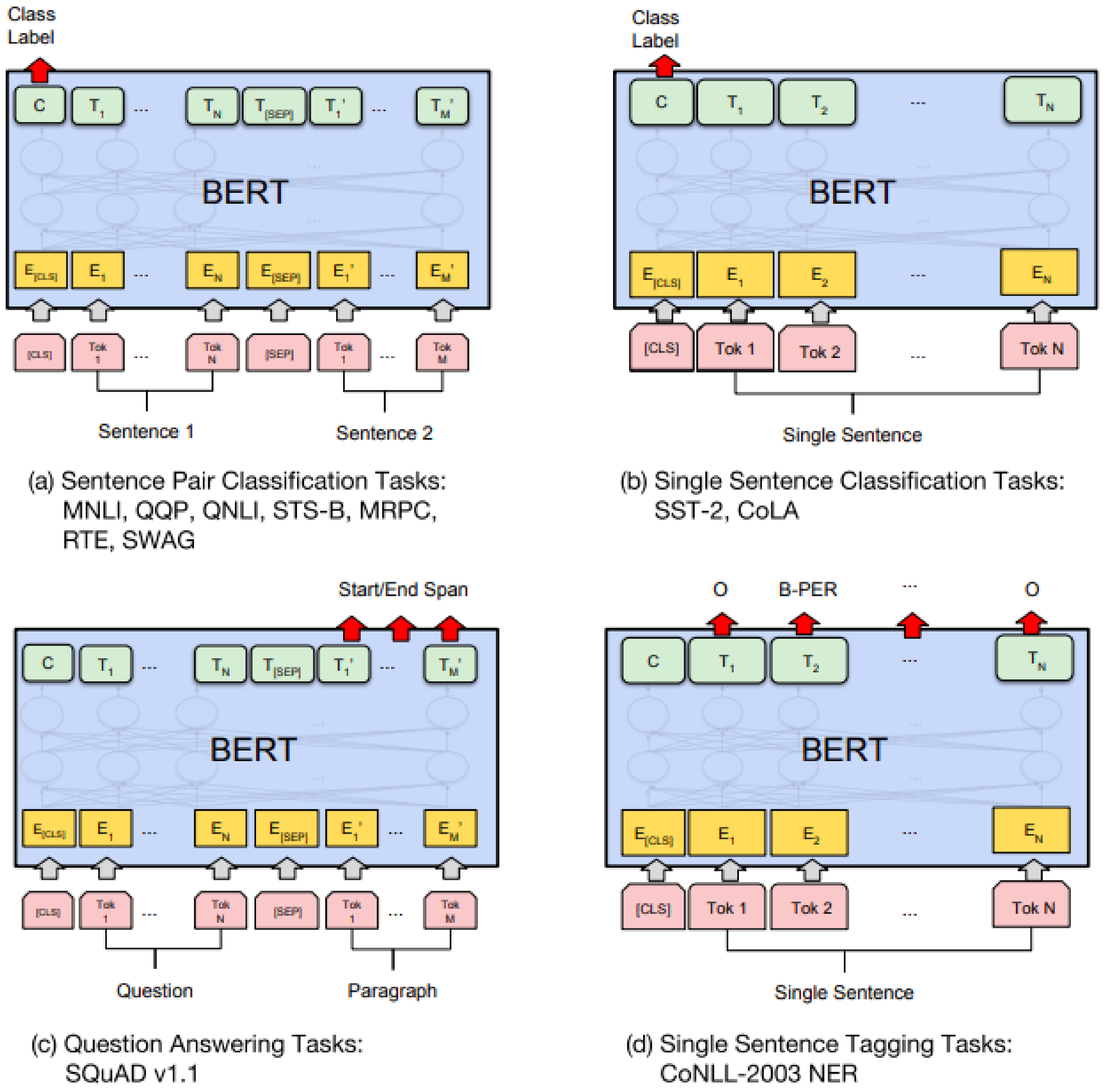


110M Parameters



340M Parameters

Task Specific Models



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

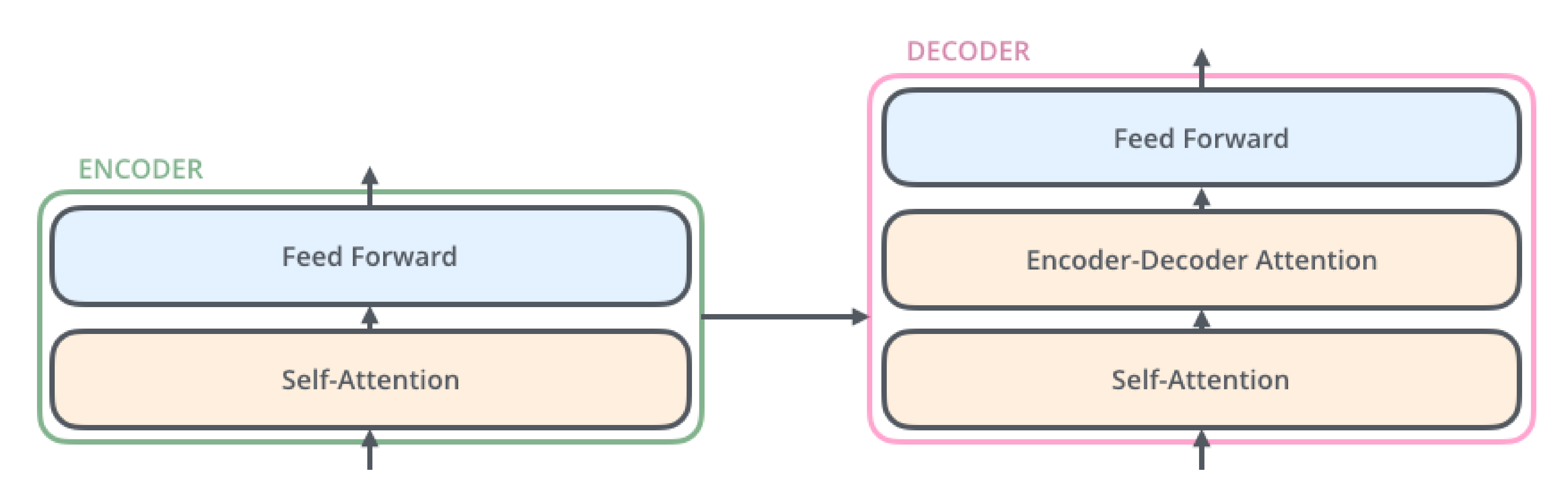
(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER t

Encoder and Decoder

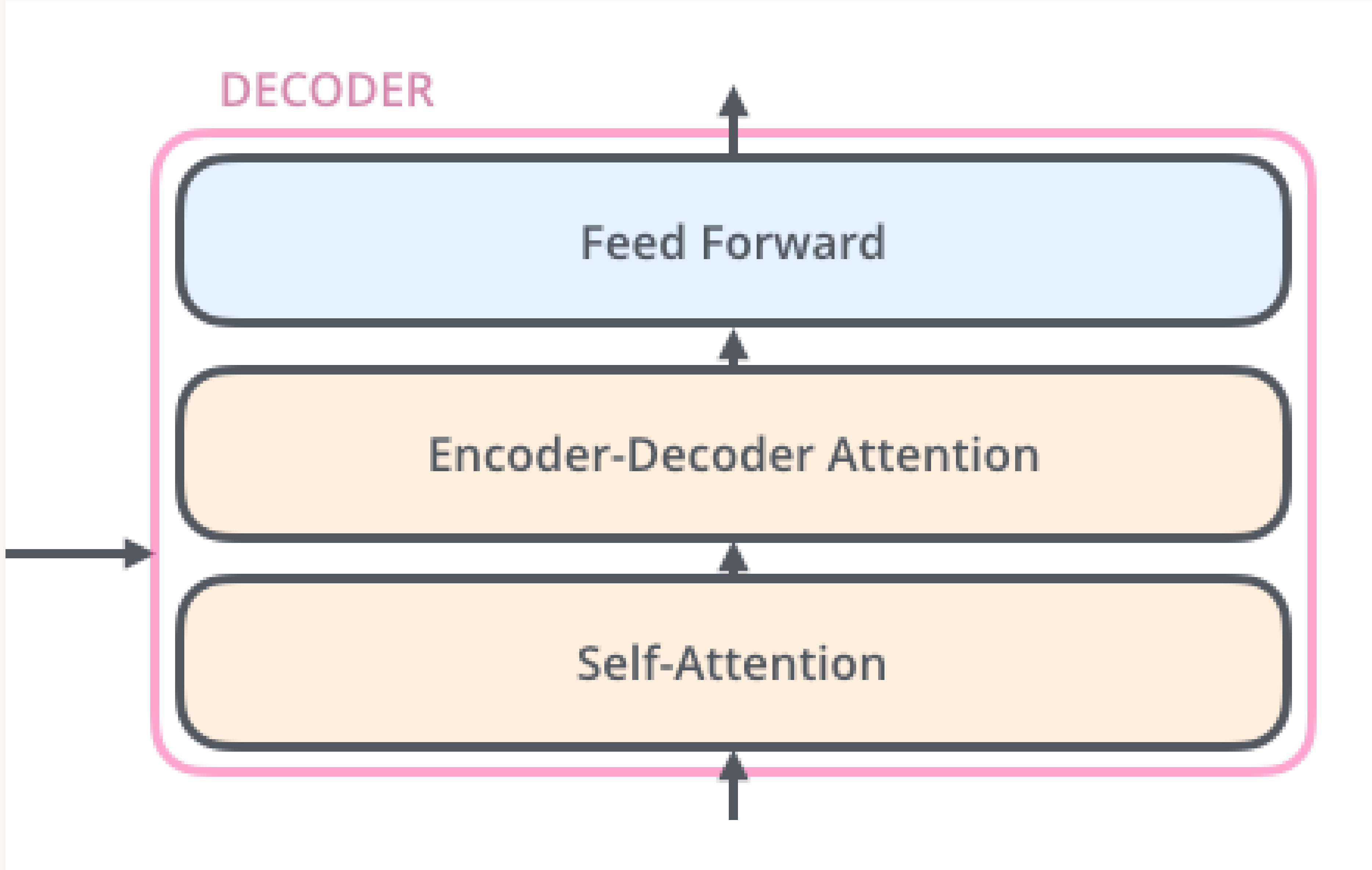
Both encoder and decoder contain a **self-attention** layer and a **feed forward** layer.

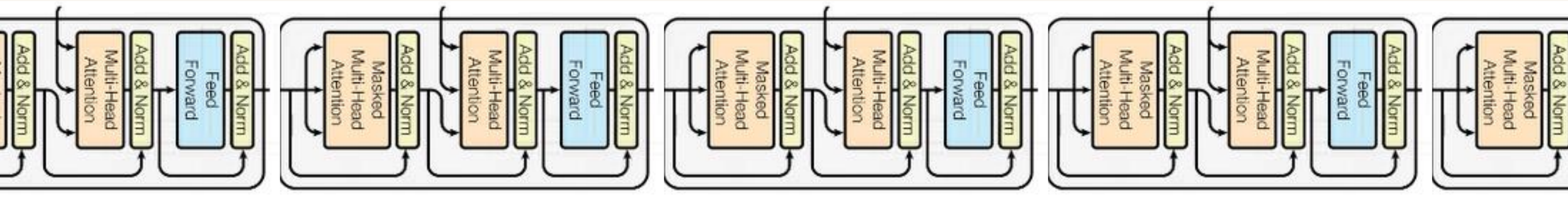
The decoder additionally contains a layer which works with both the encoder's attention values, and the decoder's attention values. This layer is called the **Encoder Decoder Attention** layer.



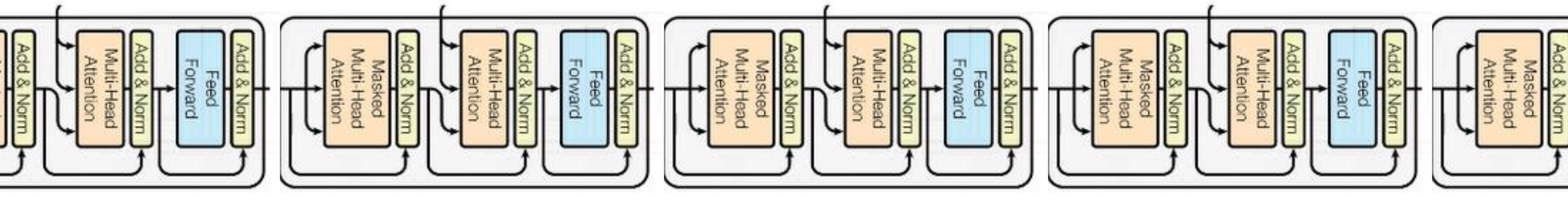
Decoder

Now, let's see what we can do with the Decoder alone!





What do we get when we stack many <u>decoders</u>, one above the other?



The G P T Architecture!

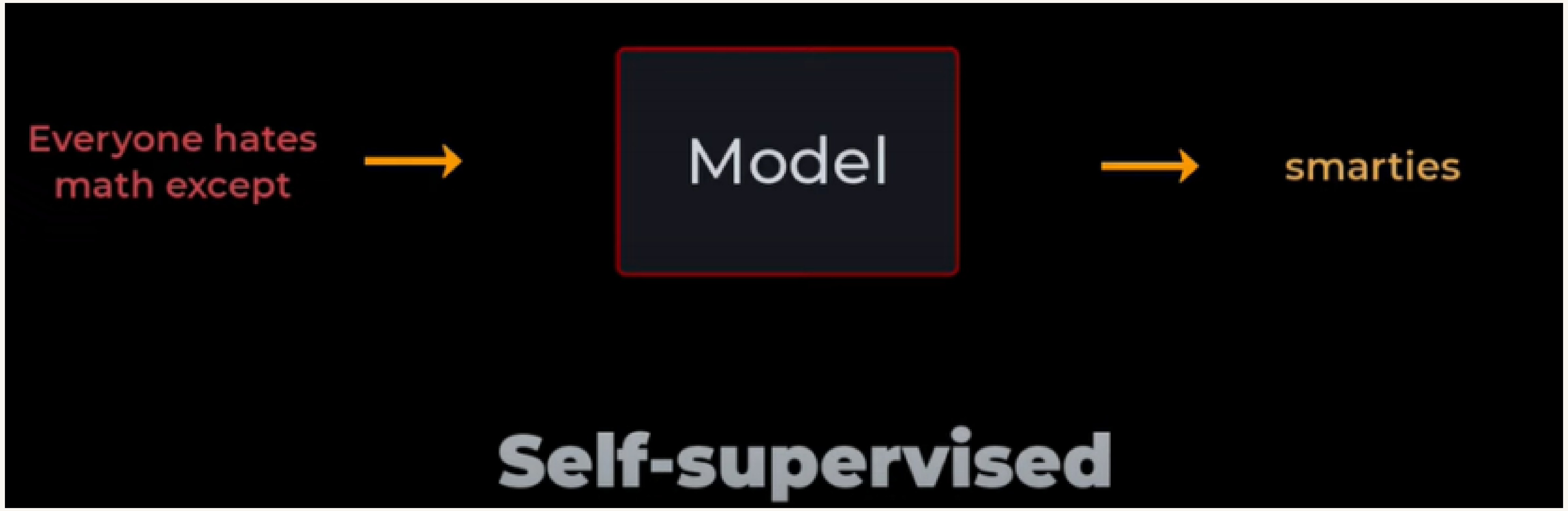
GPT

GPT

Generative Pre-training Transformers

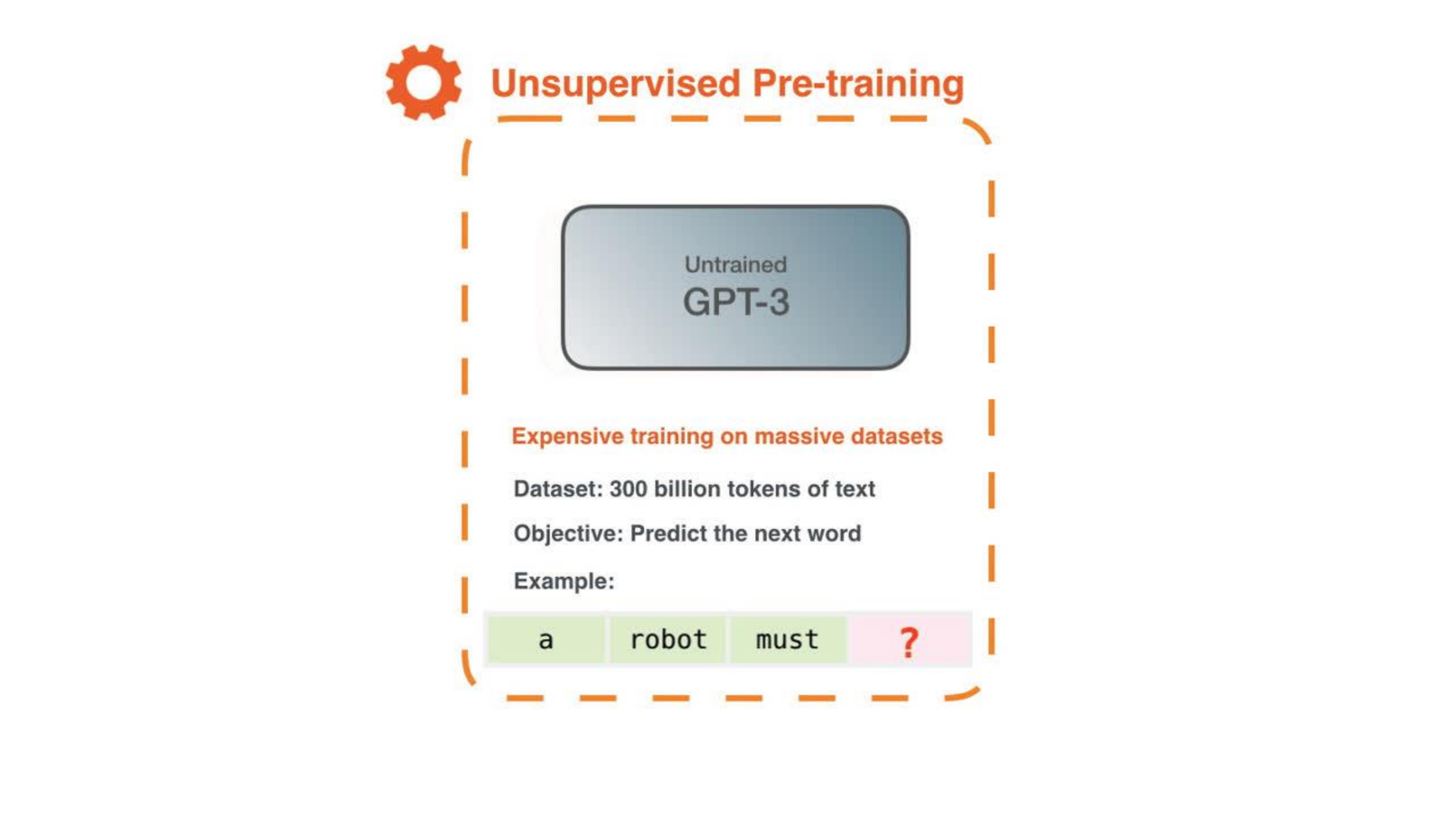
GPT

- GPT (Generative Pre-training Transformer) is a Transformer-based model designed primarily for generative tasks, meaning it generates text in an auto-regressive manner, predicting the next word based on previous ones.
- GPT models are decoder-only Transformers, focusing on generating text outputs rather than simply understanding inputs.



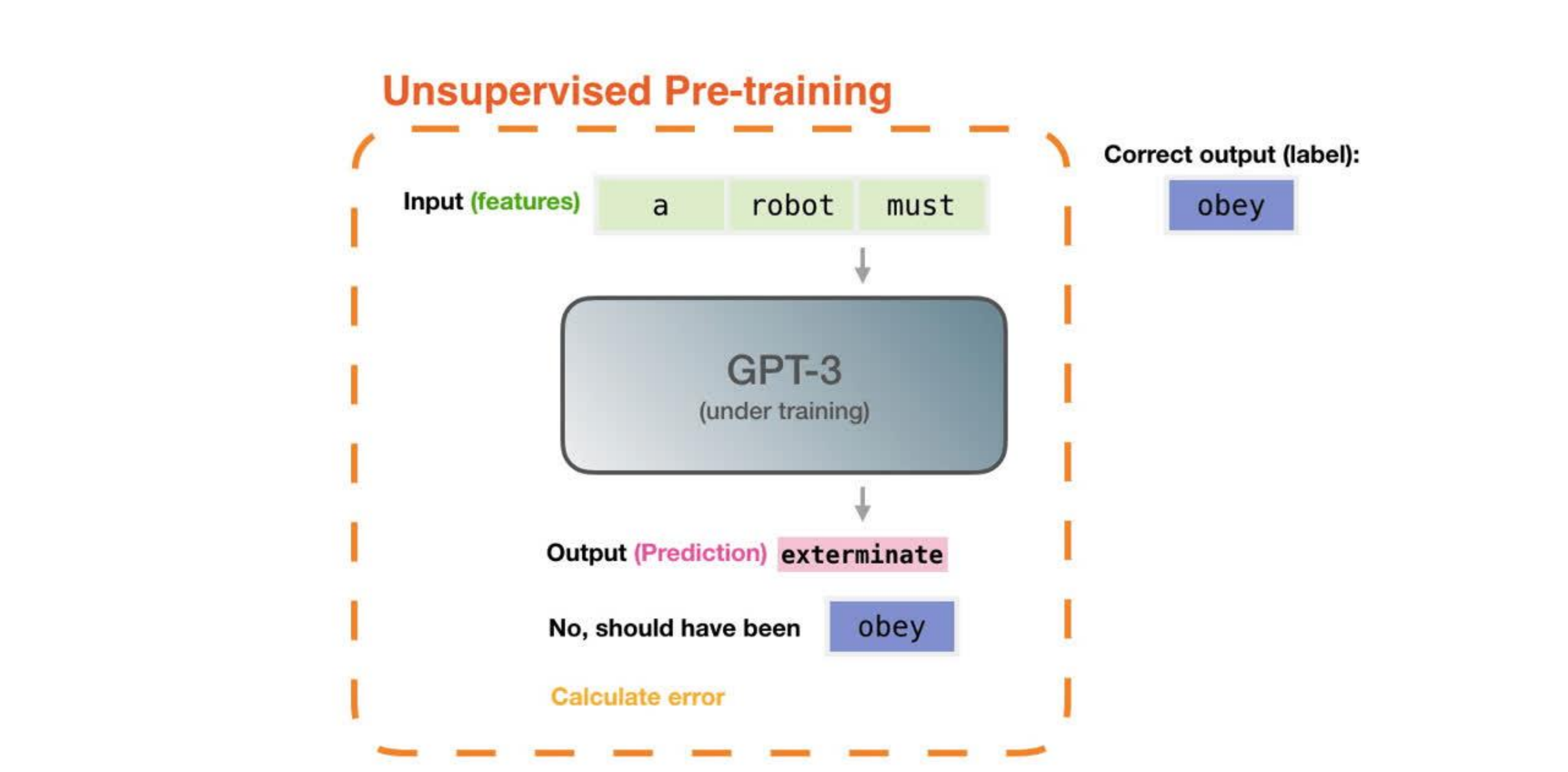
Pre Training GPT

- **Training Objective:** GPT is pre-trained on large-scale web text using a language modeling objective, where it learns to predict the next word in a sequence. After pre-training, it can be fine-tuned for specific tasks.



Pre Training GPT

Structure: GPT uses a single text sequence as input. This sequence can be processed to produce predictions using a fully connected layer after the Transformer's final hidden state.



Pre Training GPT

	Features				Labels
Example: \ position:	1	2	3	4	
1	robot	must	obey	orders	must
2	robot	must	obey	orders	obey
3	robot	must	obey	orders	orders
4	robot	must	obey	orders	<eos>

Pre Training GPT

Here’s an example of how it works

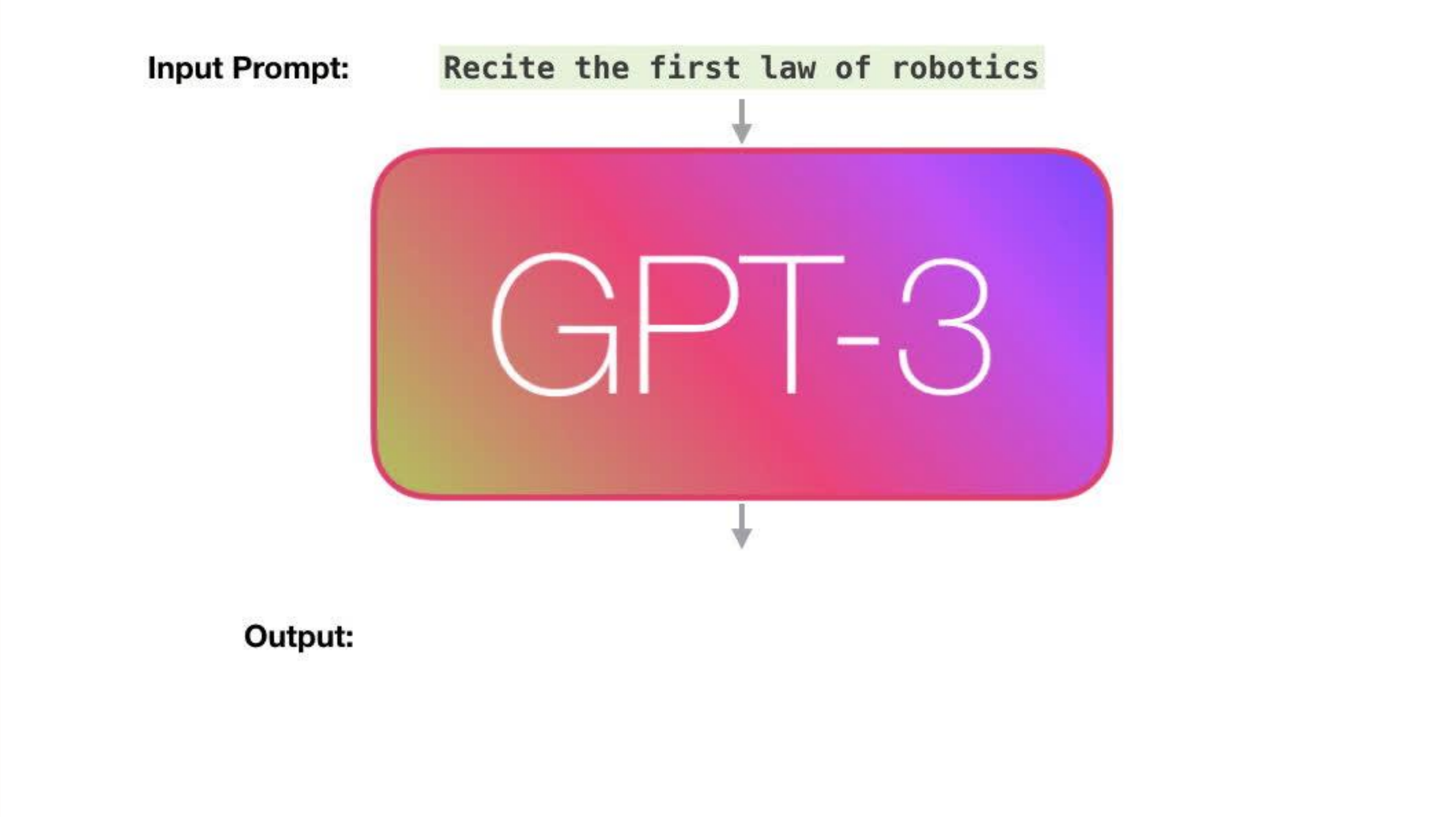
Text: Second Law of Robotics: A robot must obey the orders given it by human beings

Generated training examples

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		

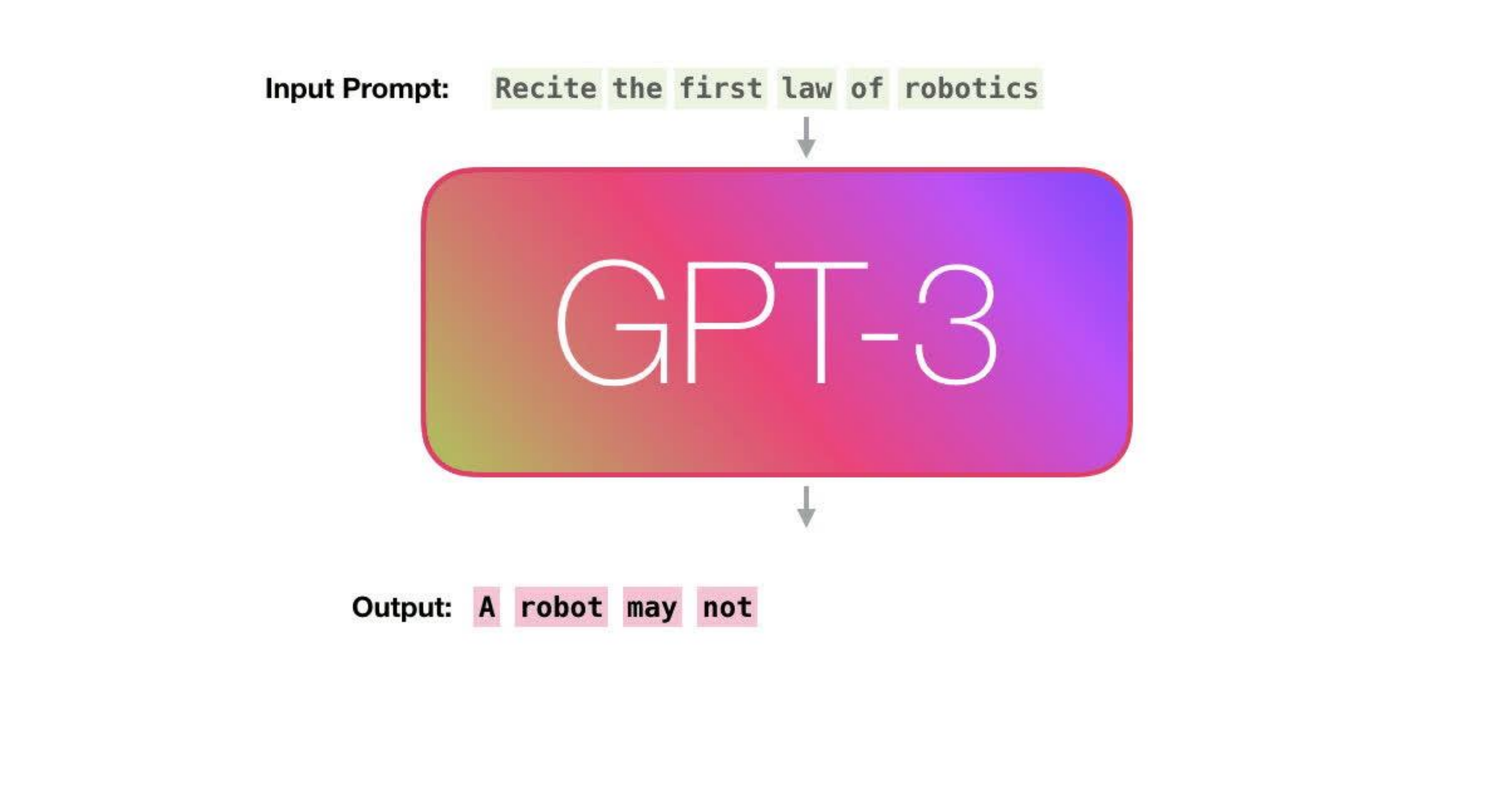
Inferencing GPT

What happens under the hood?



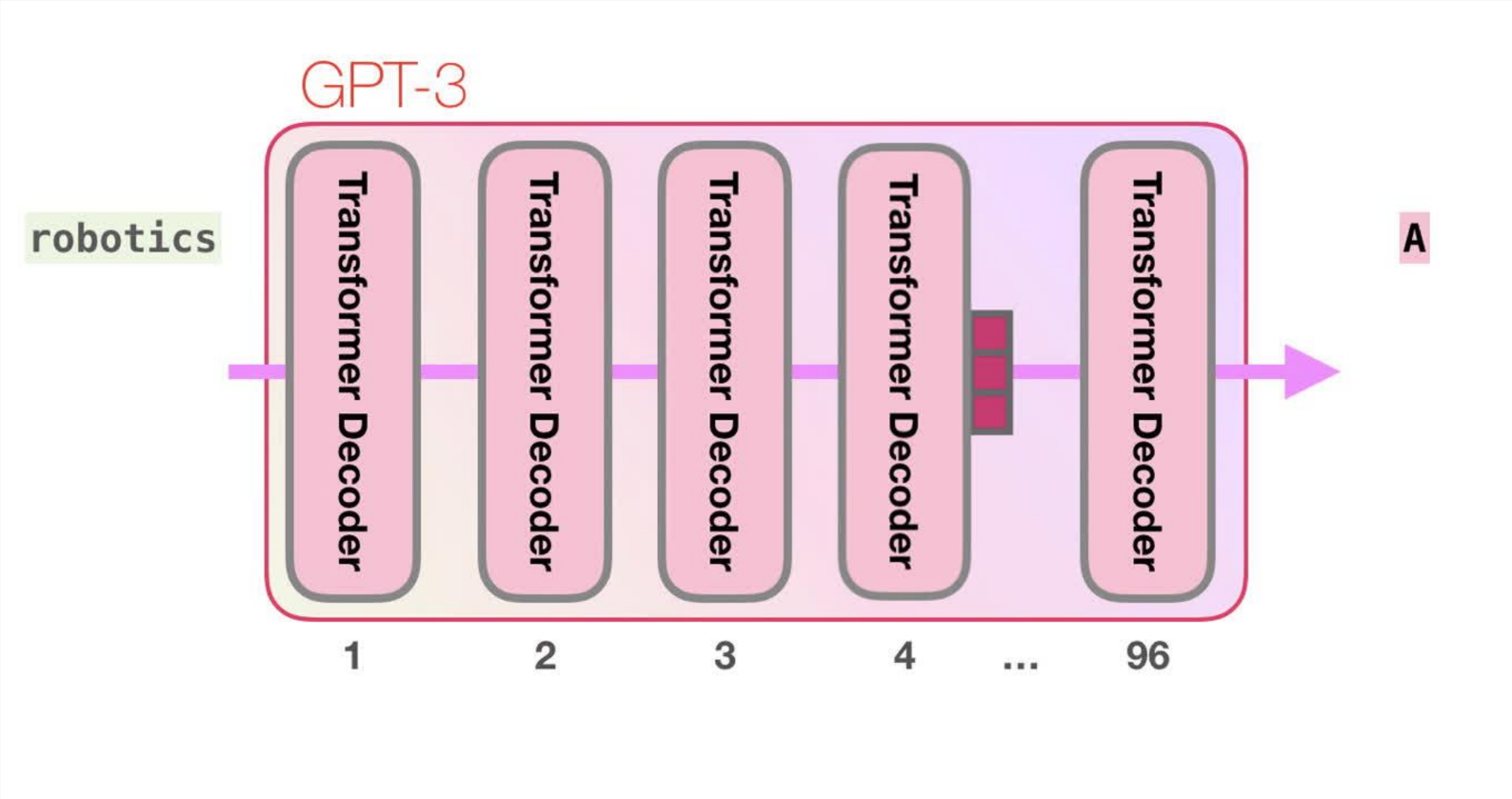
Inferencing GPT

Text is generated, word by word



Inferencing GPT

Input embeddings pass through several decoder layers



Thank You!