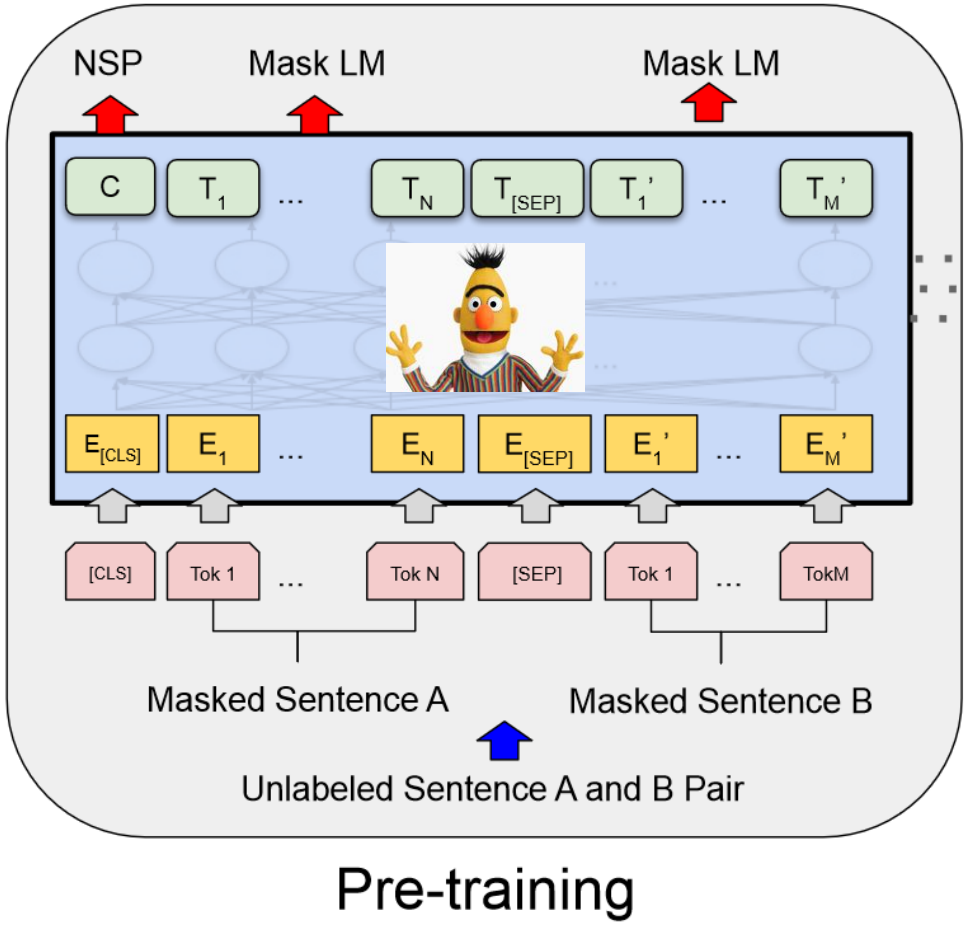


Pre-training

BERT explained from scratch

Umar Jamil

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Prerequisites

- Structure of the Transformer model and how the attention mechanism works.

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

What is a language model?

A language model is a probabilistic model that assign probabilities to sequence of words.
In practice, a language model allows us to compute the following:

We usually train a neural network to predict these probabilities. A neural network trained on a large corpora of text is known as a Large Language Model (LLM).

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

How to train a language model?

Imagine we want to train a language model on Chinese poems, for example the following one:

李白
Li Bai

English	**Chinese (simplified)**
Before my bed lies a pool of moon bright	床前明月光
I could imagine that it's frost on the ground	疑是地上霜
I look up and see the bright shining moon	举头望明月
Bowing my head I am thinking of home	低头思故乡

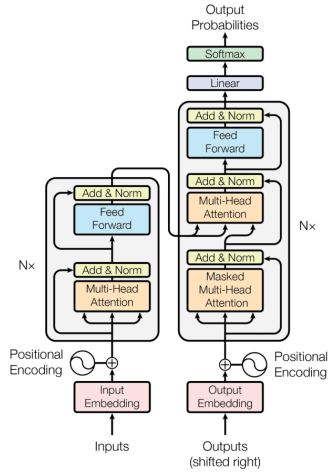
How to train a language model?

Target sequence (10 tokens) → Before my bed lies a pool of moon bright [EOS]

Cross Entropy Loss → Run **backpropagation** to update the weights

Output sequence (10 tokens) → TK1 | TK2 | TK3 | TK4 | TK5 | TK6 | TK7 | TK8 | TK9 | TK10

Neural Network
(**Transformer Encoder**)



Input sequence (10 tokens) → [SOS] Before my bed lies a pool of moon bright

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

How to inference a language model?

Imagine you're a (lazy) student who had to memorize Li Bai's poem, but only remember the first two words. How do you survive an exam?

Before my

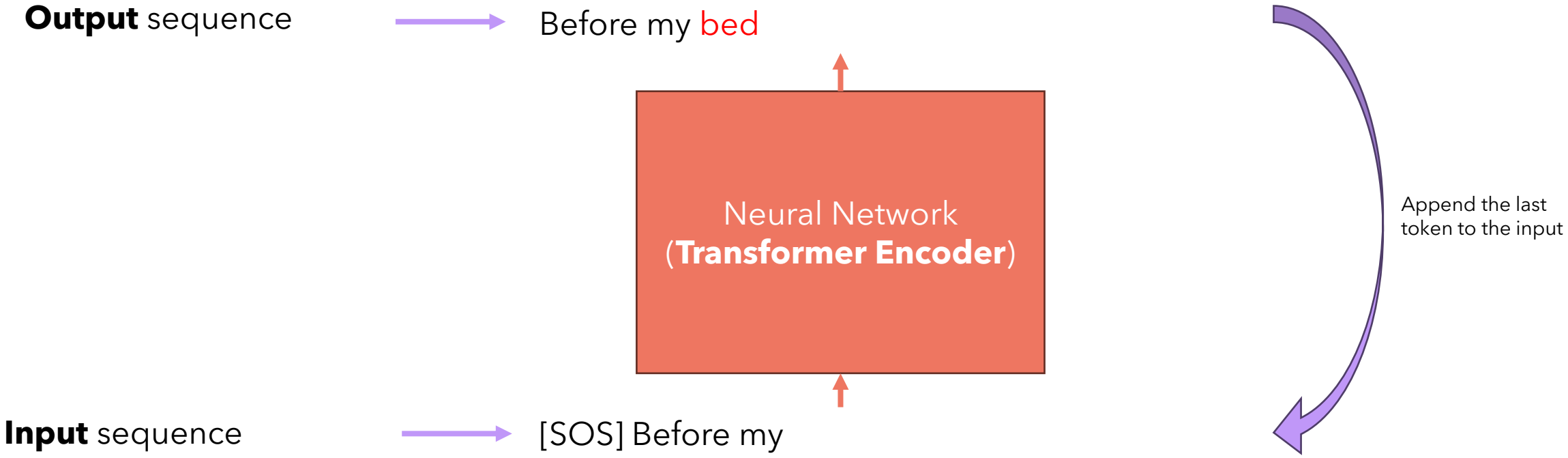
Prompt

Ask the Language Model to write the rest of the poem!

李白
Li Bai

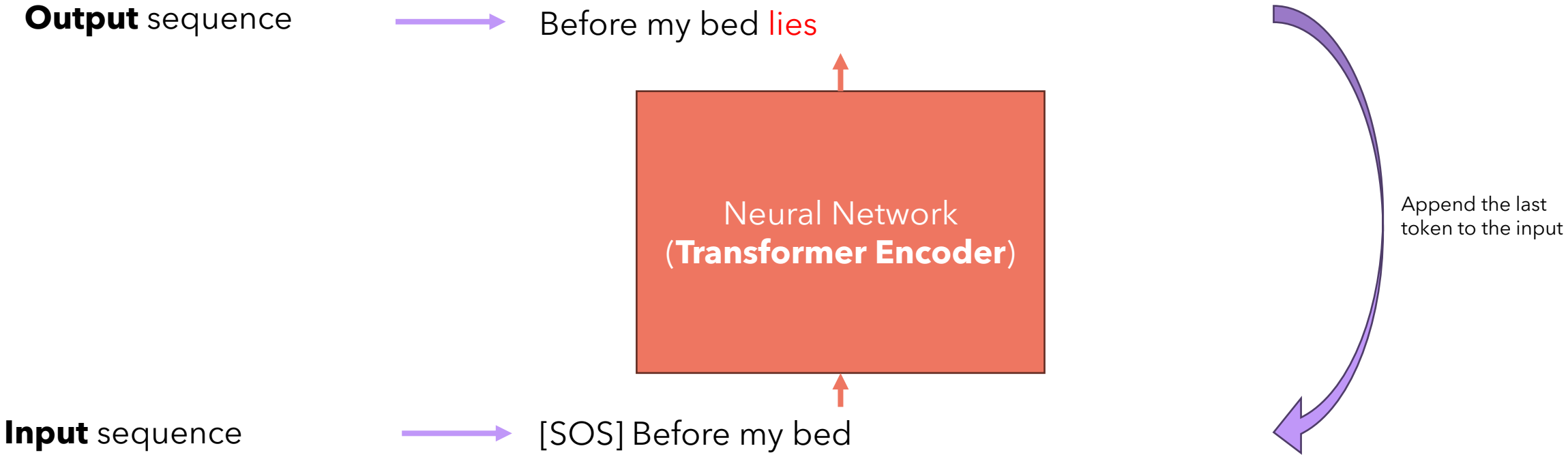
English	**Chinese (simplified)**
Before my bed lies a pool of moon bright	床前明月光
I could imagine that it's frost on the ground	疑是地上霜
I look up and see the bright shining moon	举头望明月
Bowing my head I am thinking of home	低头思故乡

How to inference a language model?

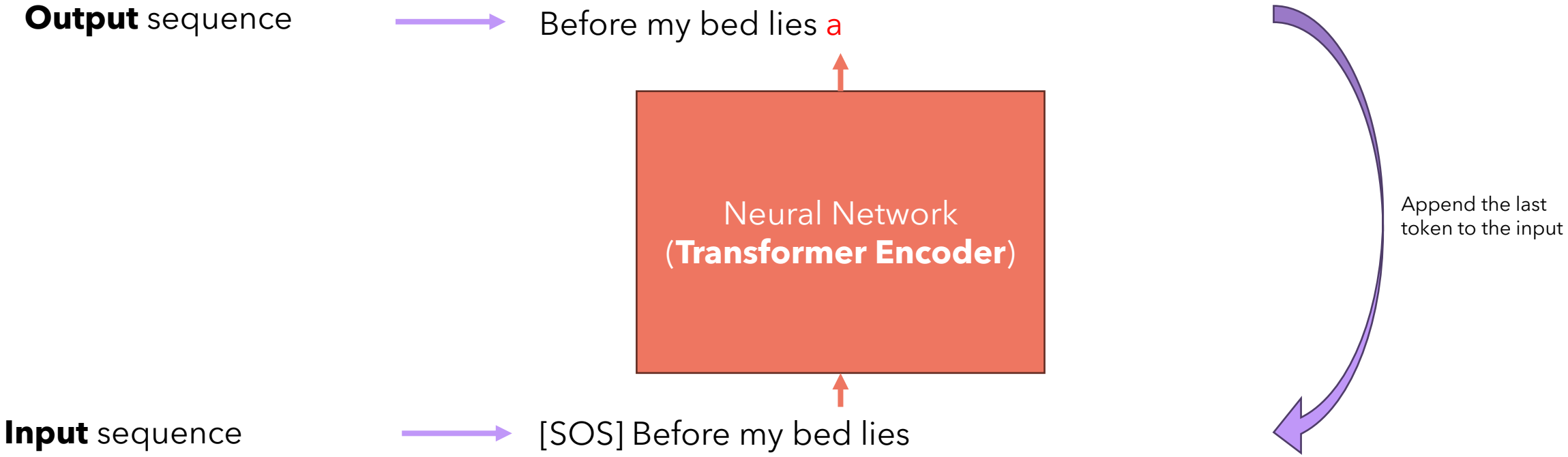


Umar Jamil – https://github.com/hkproj/bert-from-scratch

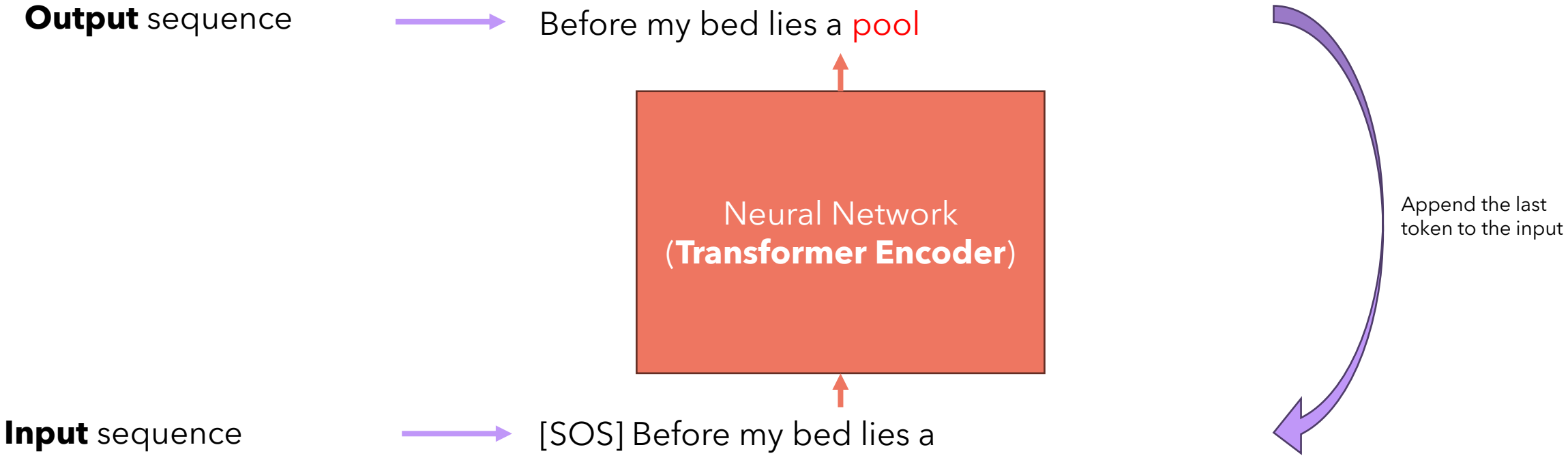
How to inference a language model?



How to inference a language model?

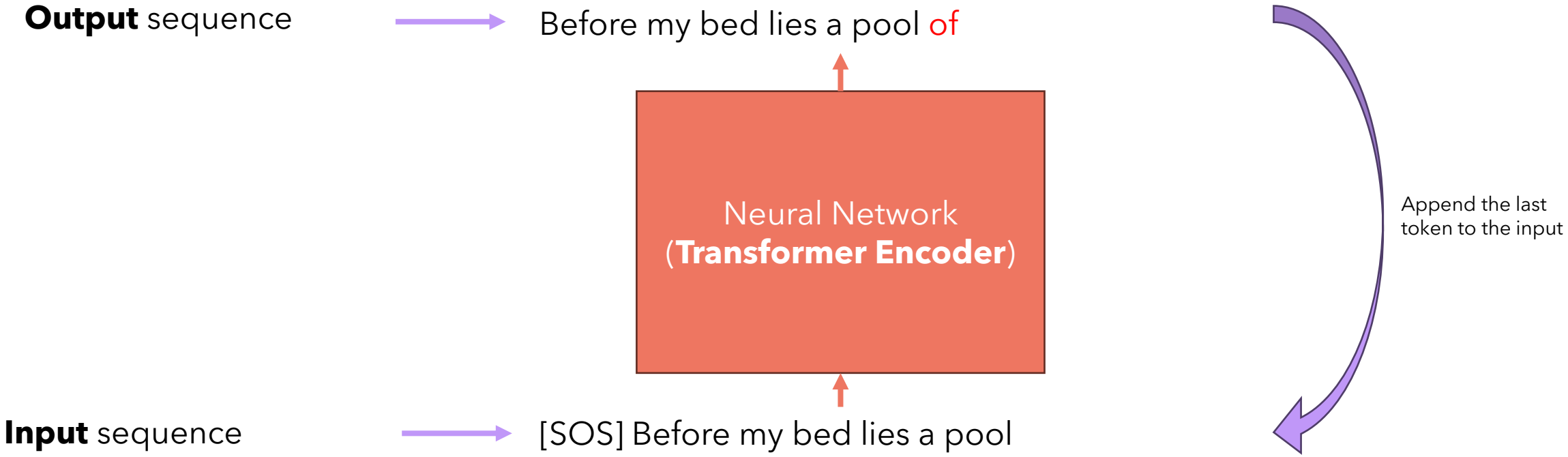


How to inference a language model?

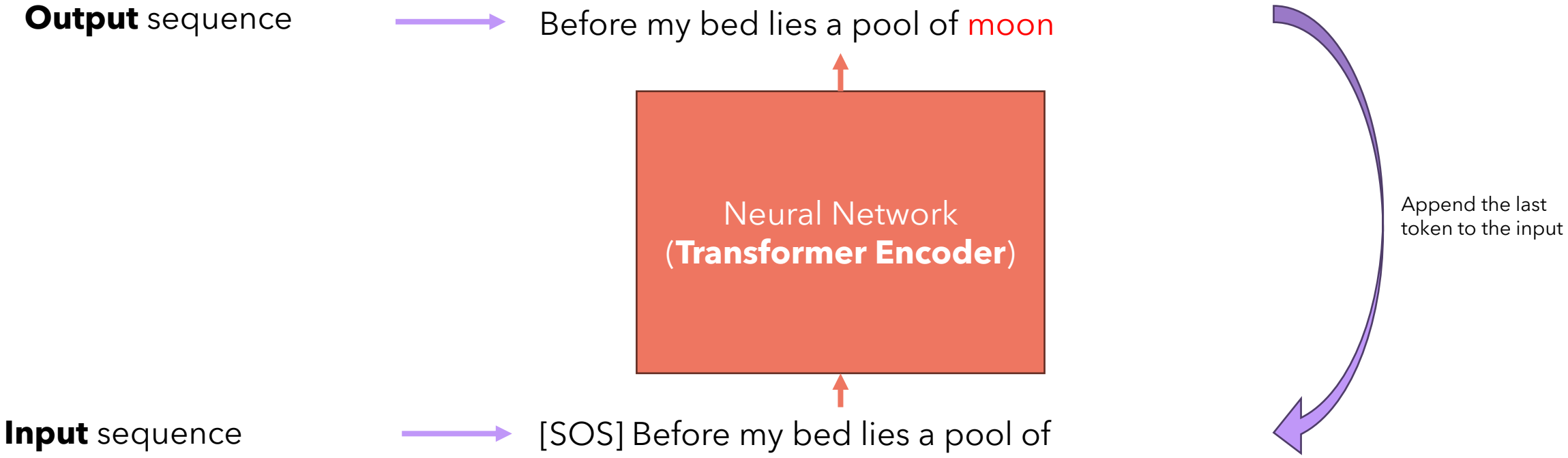


Umar Jamil – https://github.com/hkproj/bert-from-scratch

How to inference a language model?

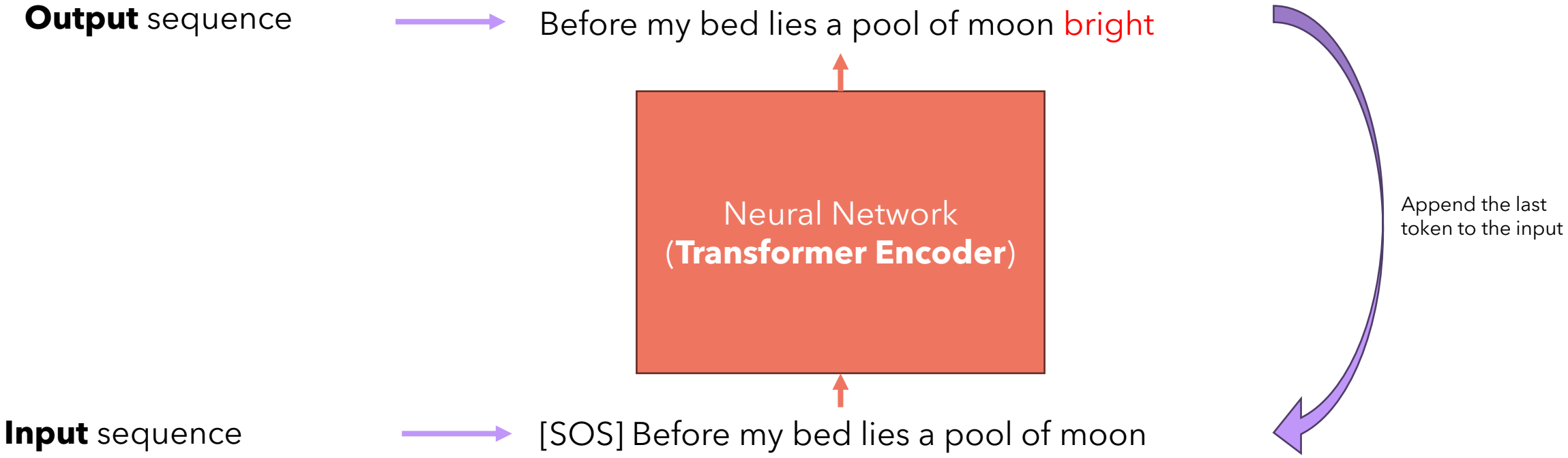


How to inference a language model?



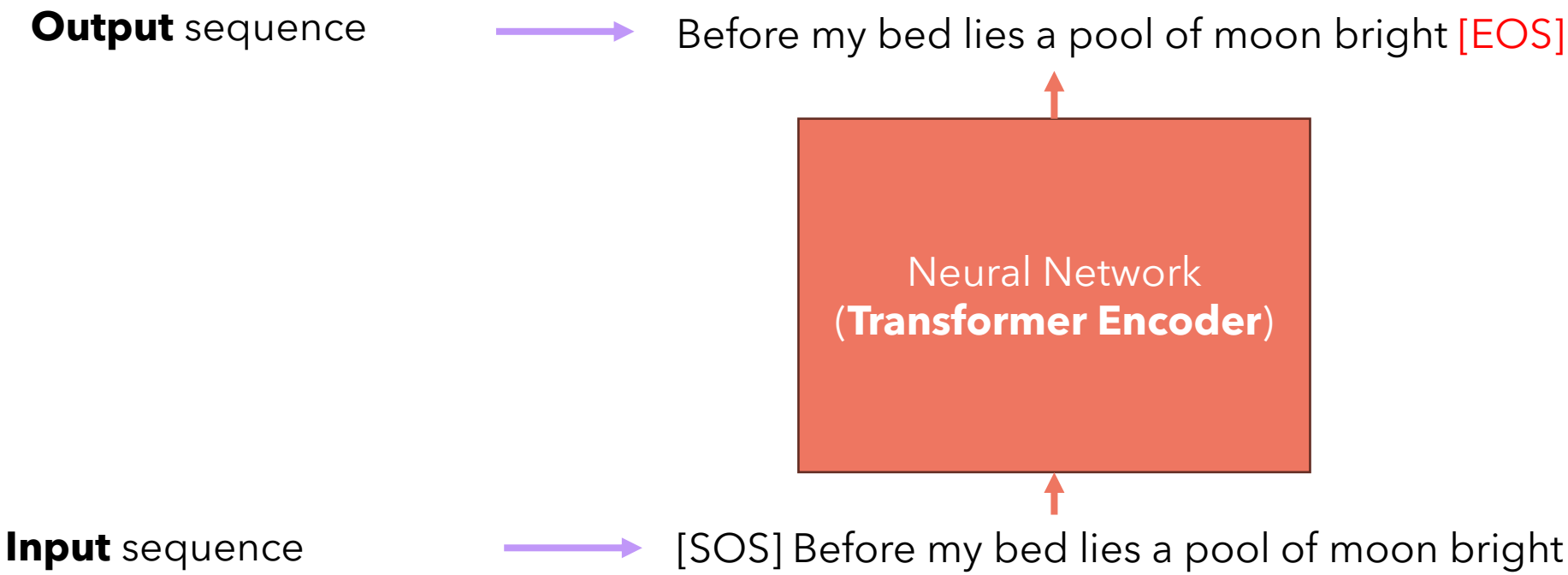
Umar Jamil – https://github.com/hkproj/bert-from-scratch

How to inference a language model?



Umar Jamil – https://github.com/hkproj/bert-from-scratch

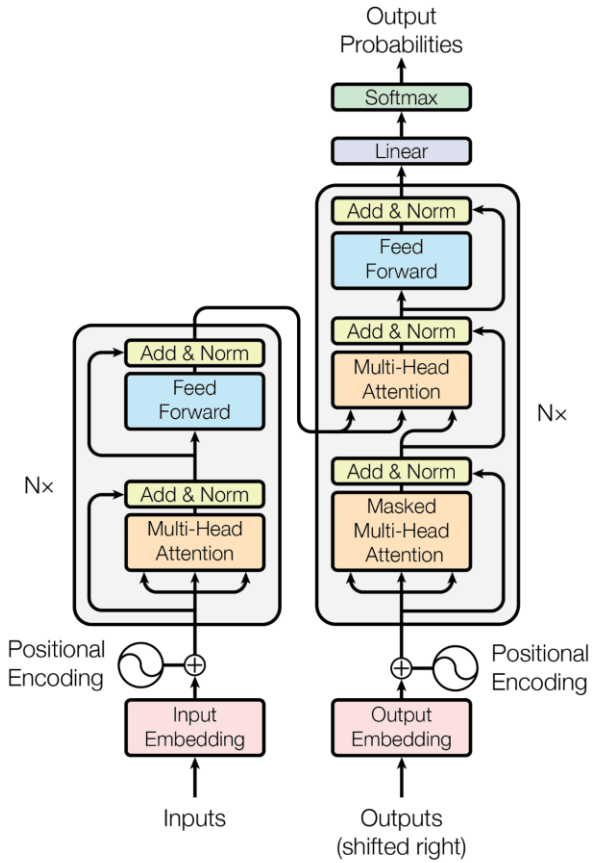
How to inference a language model?



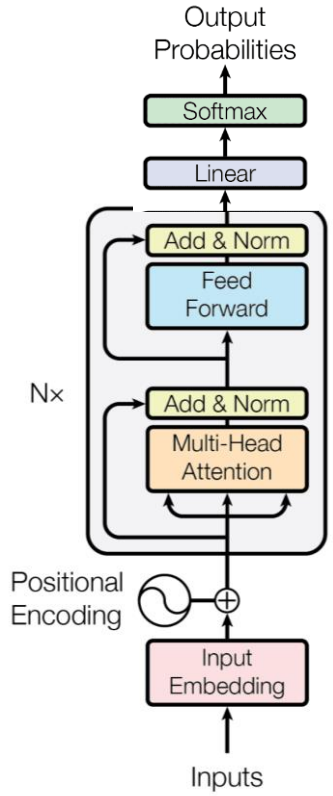
Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Transformer Encoder architecture



Transformer

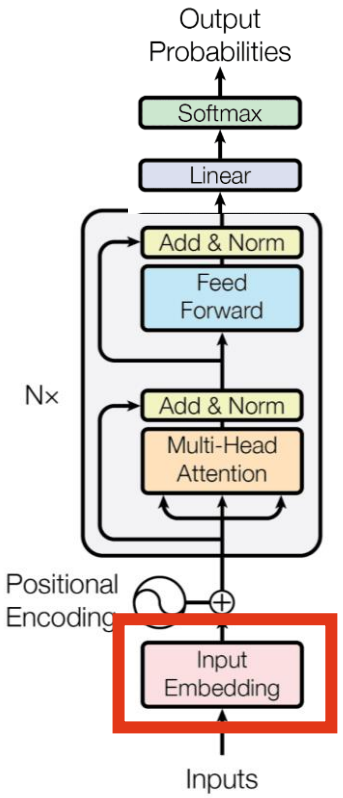


Transformer Encoder

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

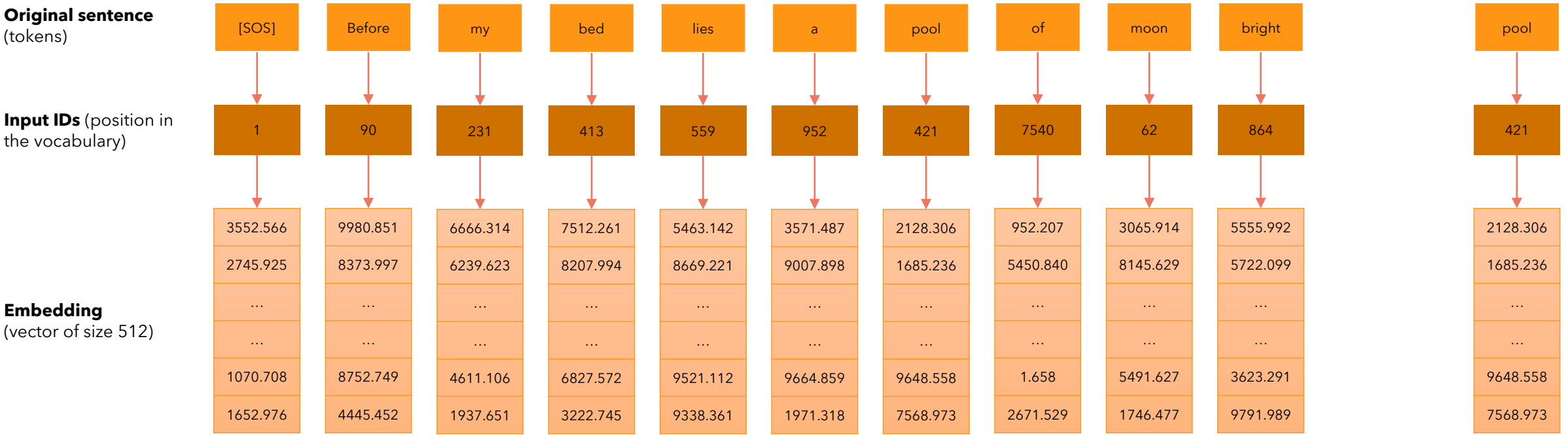
- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task



Umar Jamil – https://github.com/hkproj/bert-from-scratch

Let's convert the input into **Input Embeddings!**

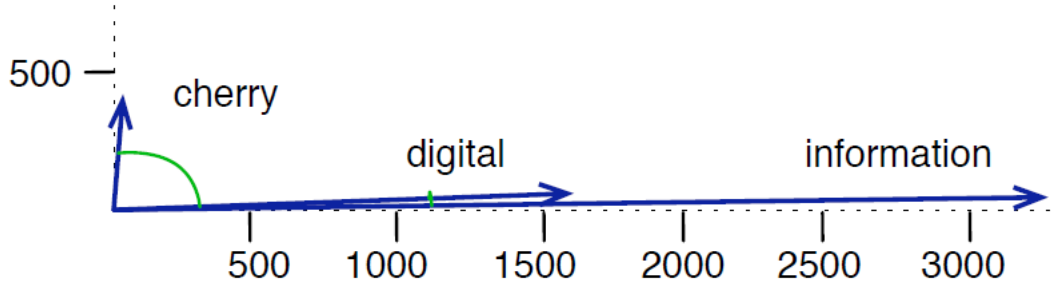
Just an example. Not in the original poem.



We define $d_{model} = 512$, which represents the size of the embedding vector of each word

Why do we use vectors to represent words?

Given the words "**cherry**", "**digital**" and "**information**", if we represent the embedding vectors using only 2 dimensions (X, Y) and we plot them, we hope to see something like this: the angle between words with similar meaning is small, while the angle between words with different meaning is big. So, the embeddings "capture" the meaning of the words they represent by projecting them into a high-dimensional space of size d_{model}.

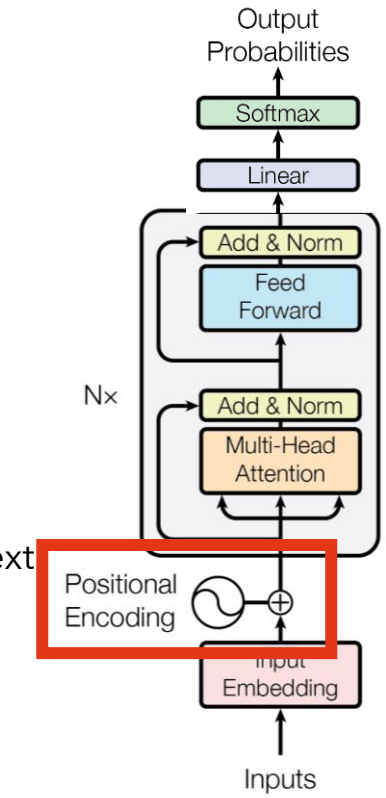


Source: Speech and Language Processing 3rd Edition Draft, Dan Jurafsky and James H. Martin

We commonly use the **cosine similarity**, which is based on the **dot product** between the two vectors.

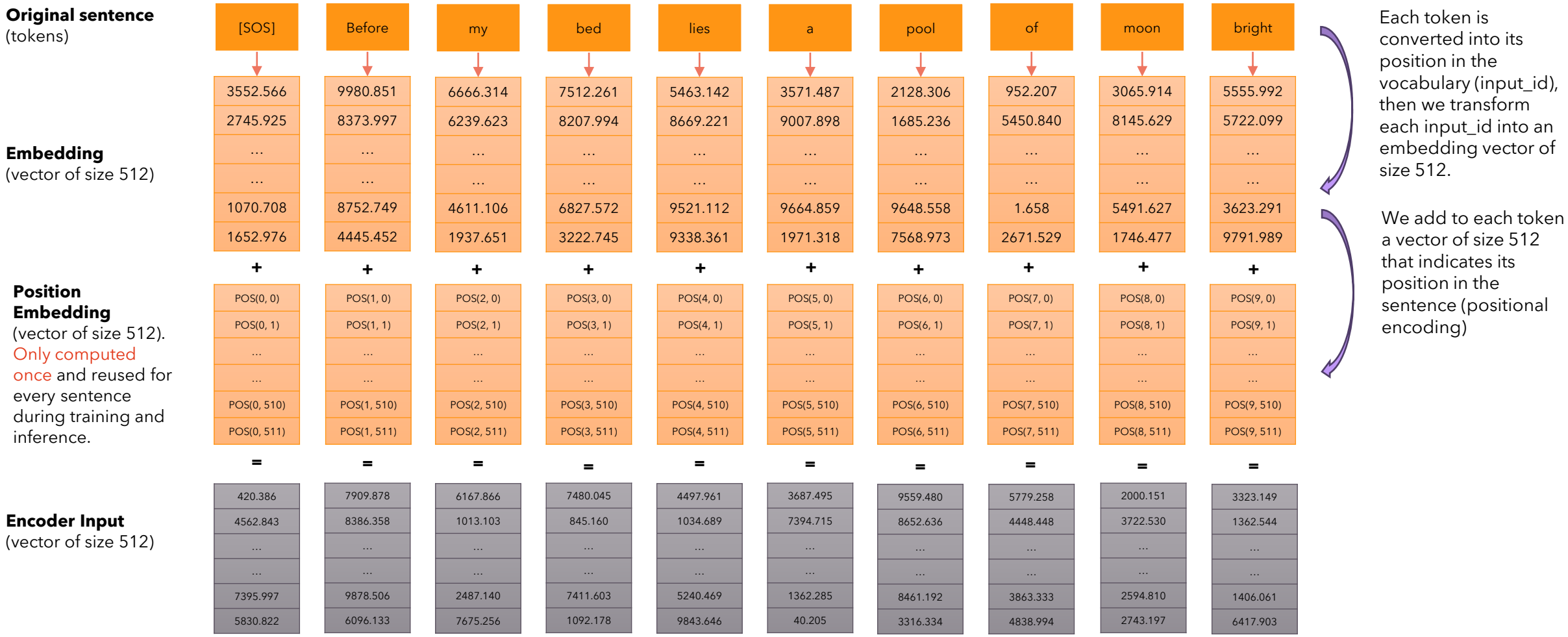
Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task



Umar Jamil – https://github.com/hkproj/bert-from-scratch

Let's add **Positional Encodings!**



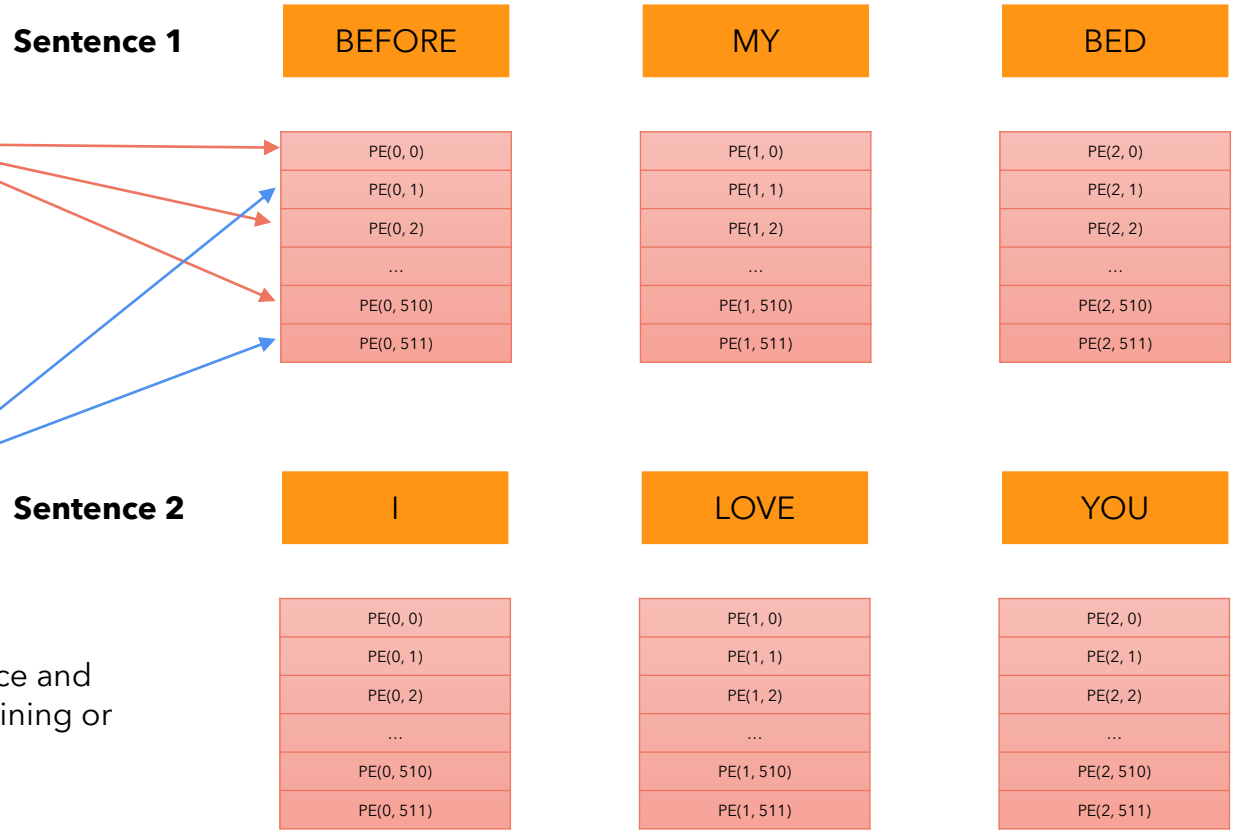
Each token is converted into its position in the vocabulary (input_id), then we transform each input_id into an embedding vector of size 512.

We add to each token a vector of size 512 that indicates its position in the sentence (positional encoding)

How to compute positional encodings?

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i+1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

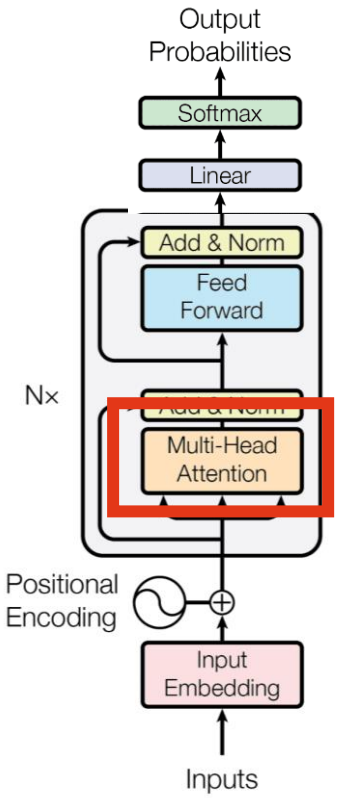


We only need to compute the positional encodings once and then reuse them for every sentence, no matter if it is training or inference.

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

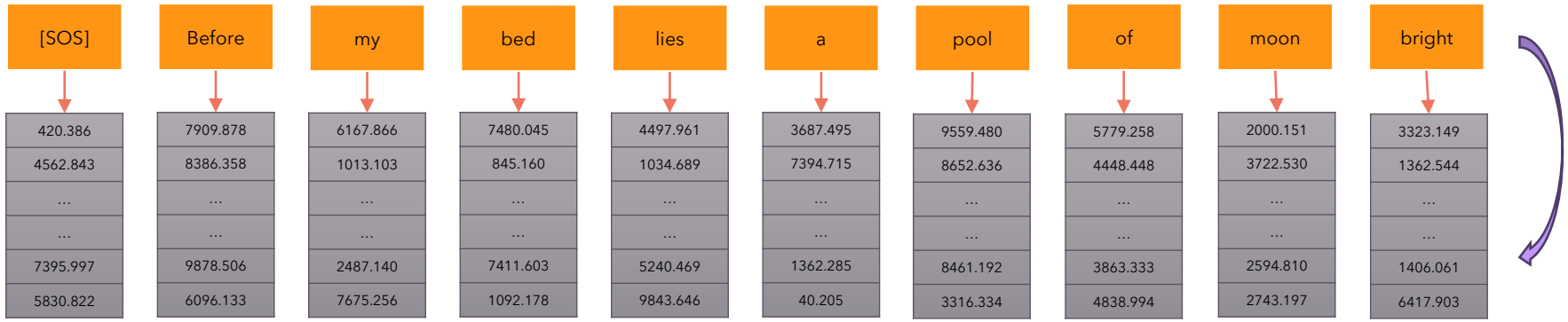


Umar Jamil – https://github.com/hkproj/bert-from-scratch

The self-attention mechanism: input

Original sentence
(tokens)

Encoder Input
(vector of size 512)



Each token is converted into its position in the vocabulary (input_id), then we transform each input_id into an embedding vector of size 512 and add its position vector (positional encoding).

[SOS]	420.386	4562.843	...	...	7395.997	5830.822
Before	7909.878	8386.358	...	...	9878.506	6096.133
my	6167.866	1013.103	...	...	2487.140	7675.256
bed	7480.045	845.160	...	...	7411.603	1092.178
lies	4497.961	1034.689	...	...	5240.469	9843.646
a	3687.495	7394.715	...	...	1362.285	40.205
pool	9559.480	8652.636	...	...	8461.192	3316.334
of	5779.258	4448.448	...	...	3863.333	4838.994
moon	2000.151	3722.530	...	...	2594.810	2743.197
bright	3323.149	1362.544	...	...	1406.061	6417.903

Matrix of shape (10, 512) where each row represents a token in the input sequence.

The self-attention mechanism: Q, K and V

In a Large Language Models (LLM) we employ the Self-Attention mechanism, which means the **Query (Q)**, **Key (K)** and **Value (V)** are the **same matrix**.

Query

[SOS]	420.386	4562.843	…	…	7395.997	5830.822
Before	7909.878	8386.358	…	…	9878.506	6096.133
my	6167.866	1013.103	…	…	2487.140	7675.256
bed	7480.045	845.160	…	…	7411.603	1092.178
lies	4497.961	1034.689	…	…	5240.469	9843.646
a	3687.495	7394.715	…	…	1362.285	40.205
pool	9559.480	8652.636	…	…	8461.192	3316.334
of	5779.258	4448.448	…	…	3863.333	4838.994
moon	2000.151	3722.530	…	…	2594.810	2743.197
bright	3323.149	1362.544	…	…	1406.061	6417.903

(10, 512)

Key

420.386	4562.843	…	…	7395.997	5830.822
7909.878	8386.358	…	…	9878.506	6096.133
6167.866	1013.103	…	…	2487.140	7675.256
7480.045	845.160	…	…	7411.603	1092.178
4497.961	1034.689	…	…	5240.469	9843.646
3687.495	7394.715	…	…	1362.285	40.205
9559.480	8652.636	…	…	8461.192	3316.334
5779.258	4448.448	…	…	3863.333	4838.994
2000.151	3722.530	…	…	2594.810	2743.197
3323.149	1362.544	…	…	1406.061	6417.903

(10, 512)

Value

420.386	4562.843	…	…	7395.997	5830.822
7909.878	8386.358	…	…	9878.506	6096.133
6167.866	1013.103	…	…	2487.140	7675.256
7480.045	845.160	…	…	7411.603	1092.178
4497.961	1034.689	…	…	5240.469	9843.646
3687.495	7394.715	…	…	1362.285	40.205
9559.480	8652.636	…	…	8461.192	3316.334
5779.258	4448.448	…	…	3863.333	4838.994
2000.151	3722.530	…	…	2594.810	2743.197
3323.149	1362.544	…	…	1406.061	6417.903

(10, 512)

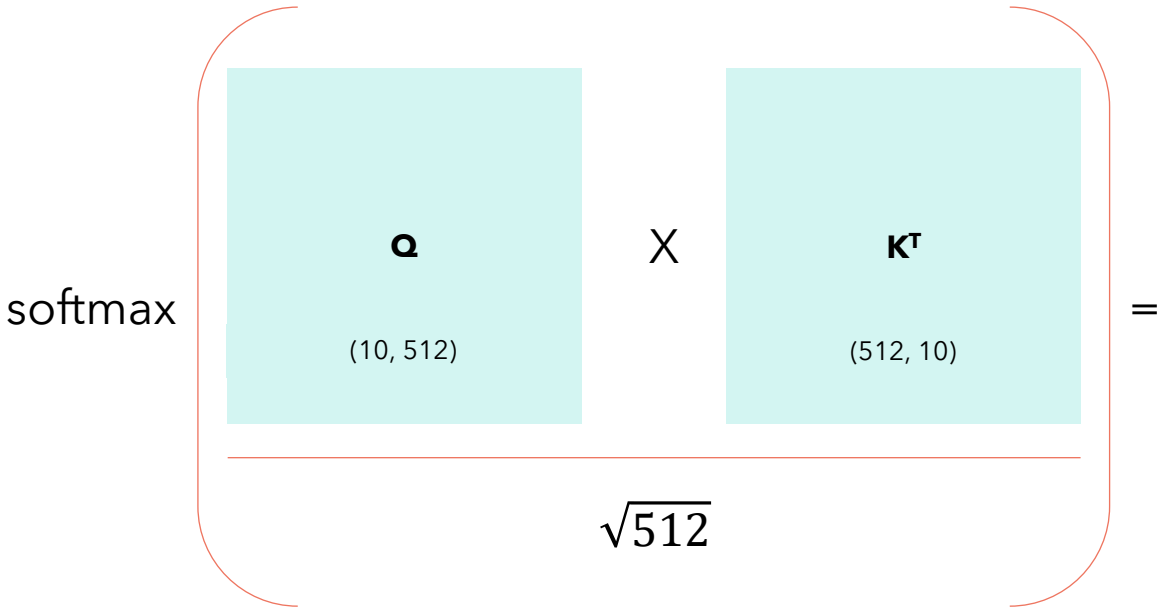
The self-attention mechanism

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Softmax of the **dot product** of the word "*my*" with the word "*bed*". Thanks to the softmax, **each row sums to 1**.

Self-Attention allows the model to relate words to each other. In our case $d_k = d_{model} = 512$.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

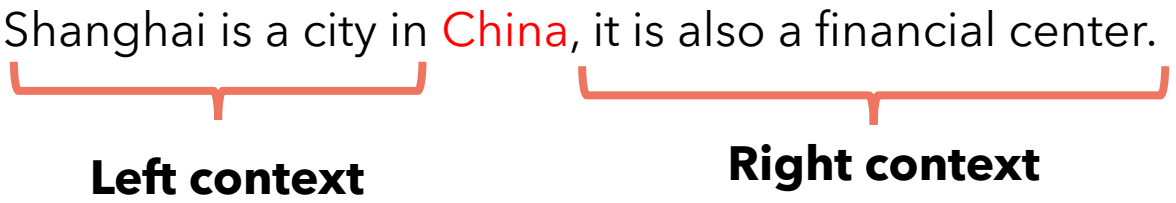


	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

The self-attention mechanism: the reason behind the causal mask

A language model is a probabilistic model that assign probabilities to sequence of words.
In practice, a language model allows us to compute the following:



To model the probability distribution above, each word should only depend on words that come **before it** (*left context*).
We will see later that in BERT we make use of both, the left and the right context.

Self-Attention mechanism: causal mask

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	**5.45**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
Before	4.28	**2.46**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
my	8.17	3.56	**5.54**	-∞	-∞	-∞	-∞	-∞	-∞	-∞
bed	6.71	4.13	6.76	**0.79**	-∞	-∞	-∞	-∞	-∞	-∞
lies	5.43	7.59	3.91	6.14	**9.03**	-∞	-∞	-∞	-∞	-∞
a	4.42	4.35	7.55	3.14	1.35	**7.57**	-∞	-∞	-∞	-∞
pool	8.36	6.00	4.56	0.52	3.13	6.78	**9.00**	-∞	-∞	-∞
of	2.21	3.72	4.16	6.30	0.66	6.14	7.46	**6.77**	-∞	-∞
moon	4.08	6.22	5.00	4.20	5.72	5.35	7.46	3.55	**4.70**	-∞
bright	6.43	8.88	6.17	3.65	4.54	5.22	5.51	5.55	0.64	**1.38**

$$\frac{QK^T}{\sqrt{d_k}}$$

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Before	0.86	**0.14**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
my	0.92	0.01	**0.07**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bed	0.47	0.04	0.49	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00
lies	0.02	0.18	0.00	0.04	**0.75**	0.00	0.00	0.00	0.00	0.00
a	0.02	0.02	0.47	0.01	0.00	**0.48**	0.00	0.00	0.00	0.00
pool	0.31	0.03	0.01	0.00	0.00	0.06	**0.59**	0.00	0.00	0.00
of	0.00	0.01	0.02	0.15	0.00	0.12	0.47	**0.23**	0.00	0.00
moon	0.02	0.16	0.05	0.02	0.10	0.07	0.55	0.01	**0.03**	0.00
bright	0.07	0.71	0.05	0.03	0.01	0.02	0.03	0.03	0.02	**0.03**

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Self-Attention mechanism: output sequence

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Before	0.86	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
my	0.92	0.01	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bed	0.47	0.04	0.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00
lies	0.02	0.18	0.00	0.04	0.75	0.00	0.00	0.00	0.00	0.00
a	0.02	0.02	0.47	0.01	0.00	0.48	0.00	0.00	0.00	0.00
pool	0.31	0.03	0.01	0.00	0.00	0.06	0.59	0.00	0.00	0.00
of	0.00	0.01	0.02	0.15	0.00	0.12	0.47	0.23	0.00	0.00
moon	0.02	0.16	0.05	0.02	0.10	0.07	0.55	0.01	0.03	0.00
bright	0.07	0.71	0.05	0.03	0.01	0.02	0.03	0.03	0.02	0.03

(10, 10)

X

V

(10, 512)

=

Attention Output

(10, 512)

Each row of the "Attention Output" matrix represents the embedding of the output sequence: it captures not only the meaning of each token, not only its position, but also the interaction of each token with all the other tokens, but only the interactions for which the softmax score is not zero. All the 512 dimensions of each vector only depend on the attention scores that are non-zero.

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Introducing BERT

BERT's architecture is made up of layers of encoders of the Transformer model:

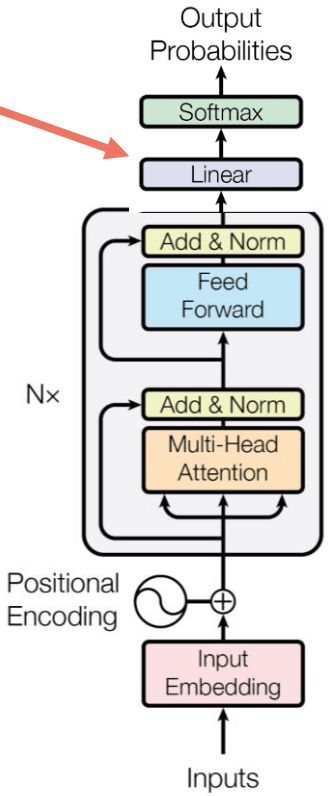
- $BERT_{BASE}$
  - 12 encoder layers
  - The size of the hidden size of the feedforward layer is 3072
  - 12 attention heads.
- $BERT_{LARGE}$
  - 24 encoder layers
  - The size of the hidden size of the feedforward layer is 4096
  - 16 attention heads.

Differences with vanilla Transformer:

- The embedding vector is 768 and 1024 for the two models
- Positional embeddings are absolute and learnt during training and limited to 512 positions
- The linear layer head changes according to the application

Uses the **WordPiece** tokenizer, which also allows sub-word tokens. The vocabulary size is ~ 30,000 tokens.

Output layer depending on the specific task



Umar Jamil – https://github.com/hkproj/bert-from-scratch

BERT vs GPT/LLaMA

BERT stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

1. Unlike common language models, BERT does not handle “special tasks” with prompts, but rather, it can be specialized on a particular task by means of fine-tuning.
2. Unlike common language models, BERT has been trained using the *left context* and the *right context*.
3. Unlike common language models, BERT is not built specifically for text generation.
4. Unlike common language models, BERT has not been trained on the Next Token Prediction task, but rather, on the Masked Language Model and Next Sentence Prediction task.

*common language models = GPT, LLaMA, etc.

Tasks in GPT/LLaMA vs BERT

Question Answering in GPT/LLaMA: **Prompt Engineering**

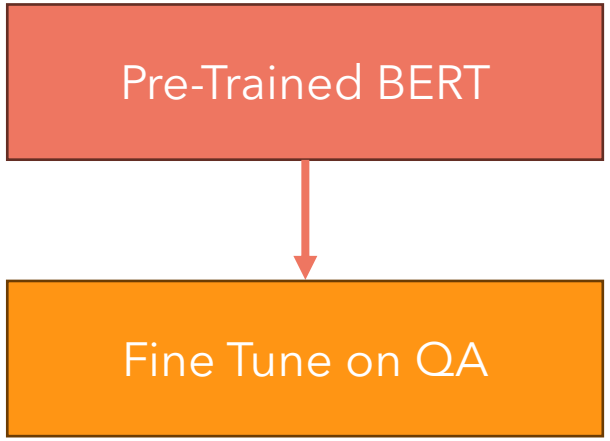
Default (GPT-3.5)

Context: “Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.”
Question: "What is the fashion capital of China? Answer with one word."
Answer: Shanghai.

Context: "Shenzhen is the tech capital of China, it is also close to Hong Kong."
Question: "What city is close to Hong Kong"?
Answer:

Shenzhen.

Question Answering in BERT: **Fine Tuning**



Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

The importance of **left** context in human conversations

Left context is used for example in phone conversations:

User: Hello! My internet line is not working, could you send a technician?

Operator: Hello! Let me check. Meanwhile, can you try restarting your WiFi router?

User: I have already restarted it but looks like the red light is not going away.

Operator: All right. I'll send someone.

The importance of **right** context in human conversations

Imagine there's a kid who just boke his mom's favorite necklace. The kid doesn't want to tell the truth to his mom, so he decides to make up a lie.

So, instead of saying directly: "Your favorite necklace has broken"

The kid may say: "Mom, I just saw the cat playing in your room and your favorite necklace has broken."

Or it may say: "Mom, aliens came through your window with laser guns and your favorite necklace has broken."

As you can see, we conditioned the lie on what we want to say next. Whatever the lie we make up, it will be always conditioned on the conclusion we want to arrive to (the necklace being broken).

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Masked Language Model (MLM)

Also known as the Cloze task. It means that randomly selected words in a sentence are masked, and the model must **predict the right word given the left and right context**.

Rome is the capital of Italy, which is why it hosts many government buildings.

Randomly select one or more tokens and replace them with the special token **[MASK]**

Rome is the [MASK] of Italy, which is why it hosts many government buildings.

capital

Left and right context in BERT

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

softmax $\left(\frac{Q \times K^T}{\sqrt{768}}\right)$ =

Q (10, 768) X K^T (768, 10)

This is the reason it is a **Bidirectional Encoder.**
Each token "attends" token to its left and tokens to its right in a sentence.

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Masked Language Model (MLM): details

Rome is the capital of Italy, which is why it hosts many government buildings.

The pre-training procedure selects 15% of the tokens from the sentence to be masked.
When a token is selected to be masked (suppose the word “capital” is selected):

- 80% of the time it is replaced with the [MASK] token → Rome is the [MASK] of Italy, which is why it hosts many government buildings.
- 10% of the time it is replaced with a random token → Rome is the zebra of Italy, which is why it hosts many government buildings.
- 10% of the time it is not replaced → Rome is the capital of Italy, which is why it hosts many government buildings.

Masked Language Model (MLM): training

Target (1 token):

capital

Loss → Run **backpropagation** to update the weights

Output (14 tokens):

Input (14 tokens):

Rome is the [mask] of Italy, which is why it hosts many government buildings.

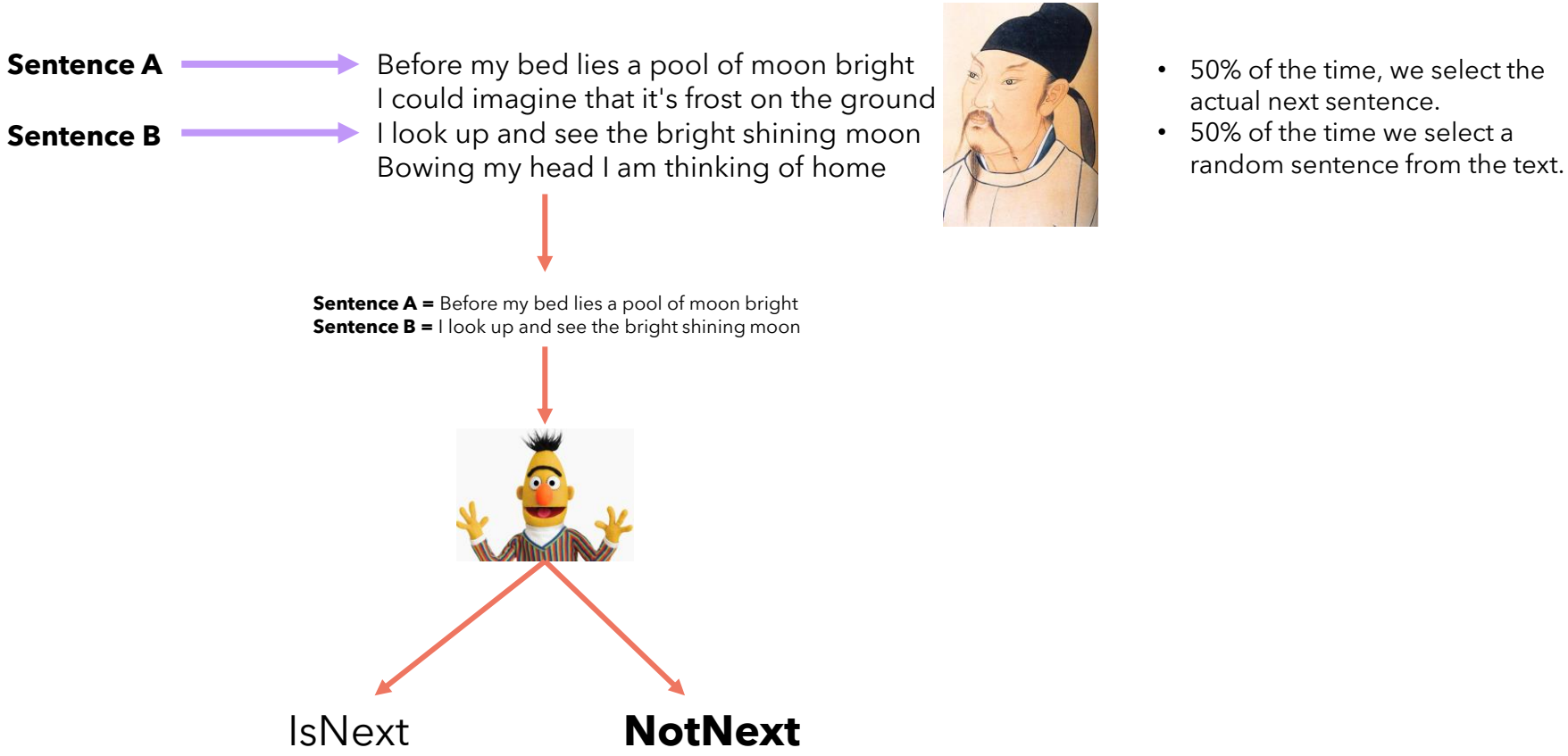
Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Next Sentence Prediction (NSP)

Many downstream applications (for example choosing the right answer given 4 choices) require learning relationships between sentences rather than single tokens, that's why BERT has been pre-trained on the Next Sentence Prediction task.

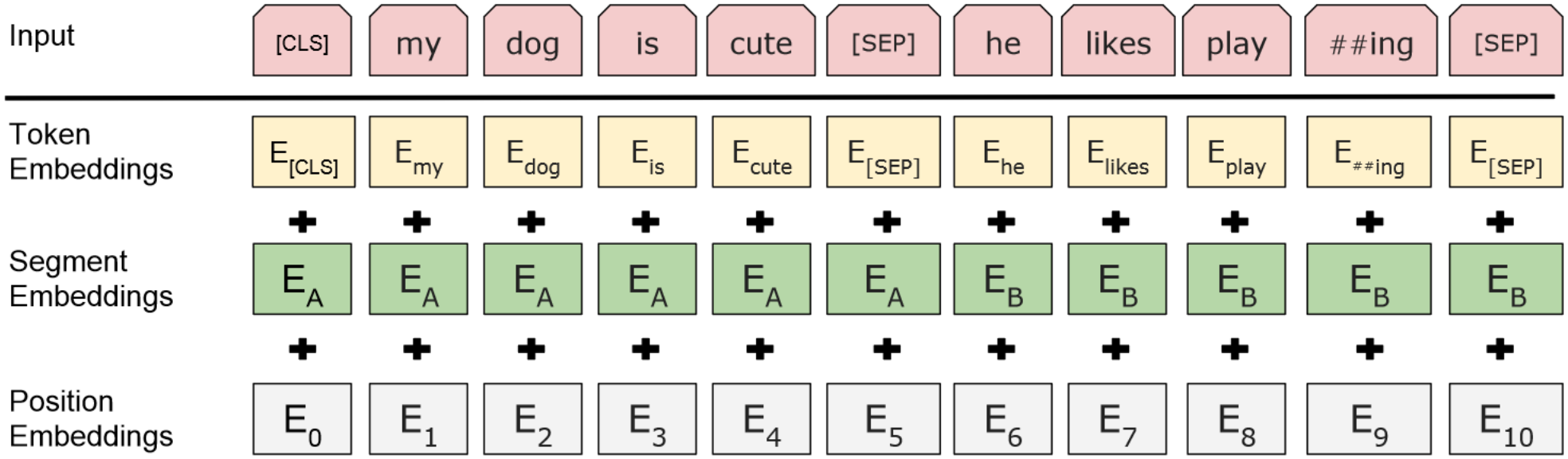


- 50% of the time, we select the actual next sentence.
- 50% of the time we select a random sentence from the text.

Next Sentence Prediction (NSP): segmentation embedding

Given the sentence A and the sentence B, how can BERT understand which tokens belongs to the sentence A and which to the sentence B? We introduce the segmentation embeddings!

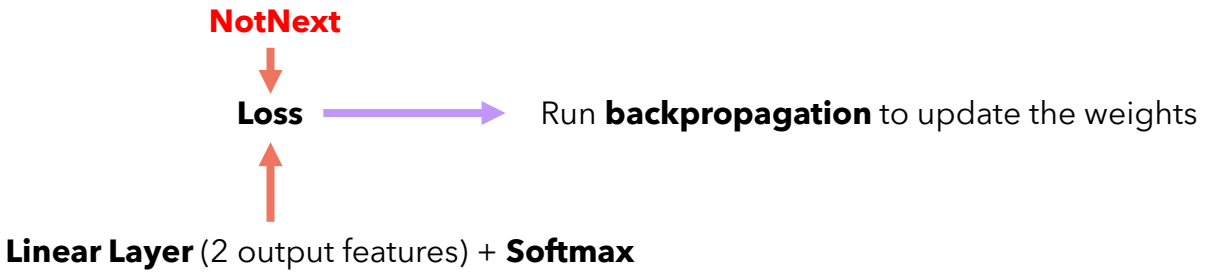
We also introduce two special tokens: **[CLS]** and **[SEP]**



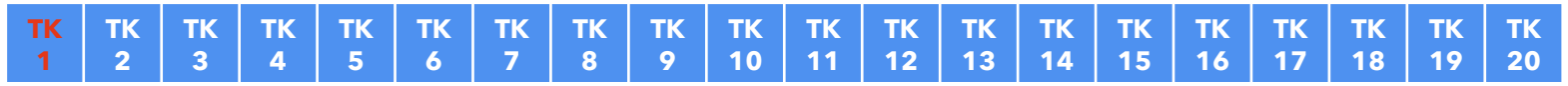
Umar Jamil – https://github.com/hkproj/bert-from-scratch

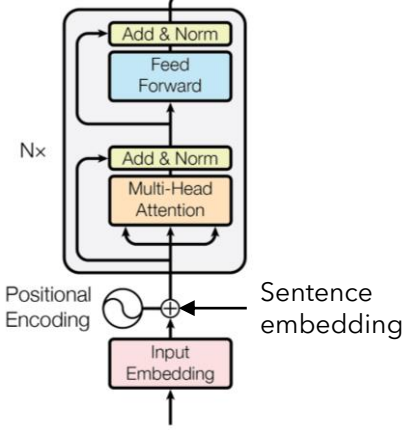
Next Sentence Prediction (NSP): training

Target (1 token):



Output (20 tokens):





Before my bed lies a pool of moon bright
I could imagine that it's frost on the ground
I look up and see the bright shining moon
Bowing my head I am thinking of home

Input (20 tokens):

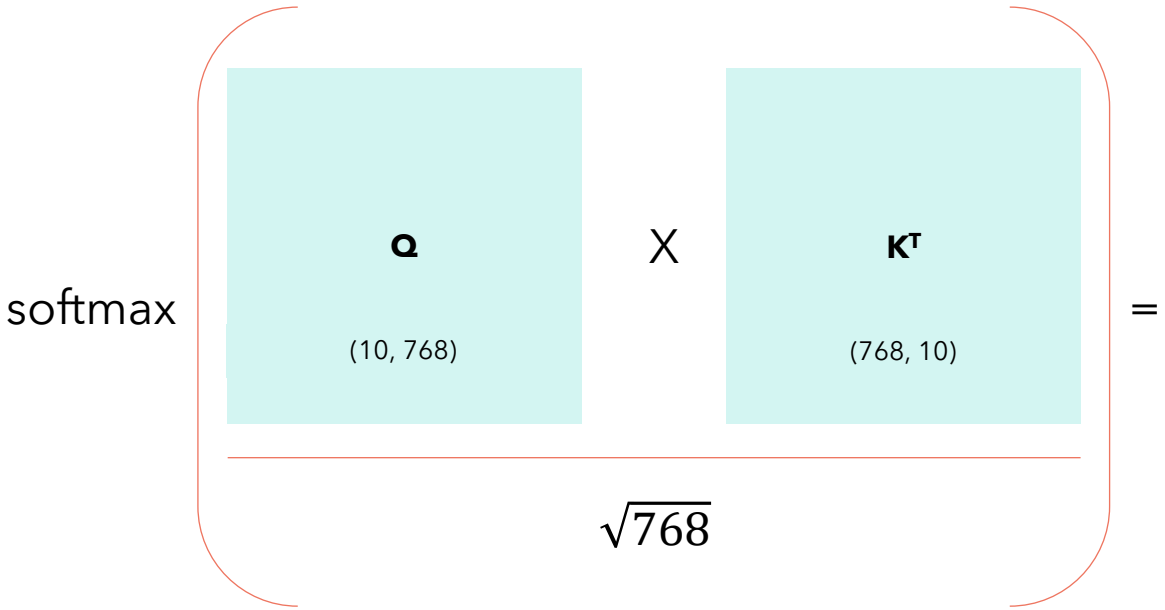
[CLS] Before my bed lies a pool of moon bright [SEP] I look up and see the bright shining moon

Sentence A **Sentence B**

Umar Jamil – https://github.com/hkproj/bert-from-scratch

[CLS] token in BERT

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



The **[CLS]** token always interacts with all the other tokens, as we do not use any mask.
So, we can consider the **[CLS]** token as a token that "captures" the information from all the other tokens.

	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

Umar Jamil – https://github.com/hkproj/bert-from-scratch

[CLS] token: output sequence

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

X

V

(10, 768)

=

Attention Output

(10, 768)

Each row of the "Attention Output" matrix represents the embedding of the output sequence: it captures not only the meaning of each token, not only its position, but also the interaction of each token with all the other tokens, but only the interactions for which the softmax score is not zero. All the 512 dimensions of each vector only depend on the attention scores that are non-zero.

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Fine-Tuning BERT

Pre-Trained BERT

Fine Tune on text classification

Fine Tune on question answering (QA)

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Text Classification

Text classification is the task of assigning a label to a piece of text. For example imagine we are running an internet provider and we receive complaints from our customers. We may want to classify requests coming from users as hardware problems, software problems or billing issues.

My router's led is not working, I tried changing the power socket but still nothing.

Hardware

My router's web page doesn't allow me to change password anymore… I tried restarting it but nothing.

Software

In this month's bill I have been charged 100$ instead of the usual 60$, why is that?

Billing

Text Classification: training

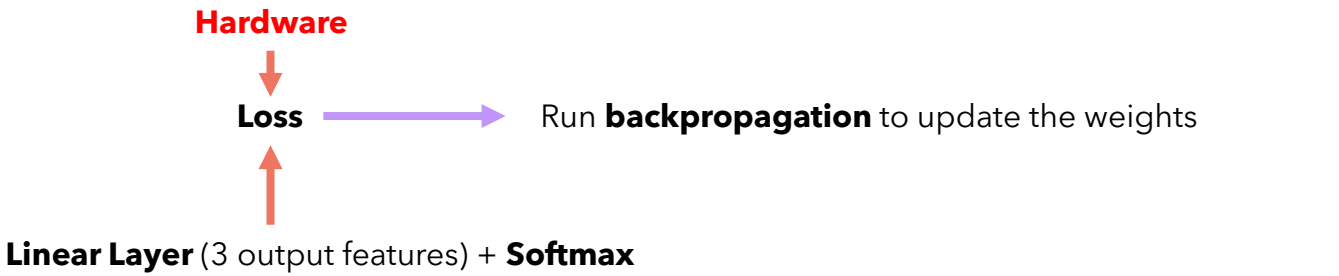
My router's led is not working, I tried changing the power socket but still nothing.

Hardware Software Billing

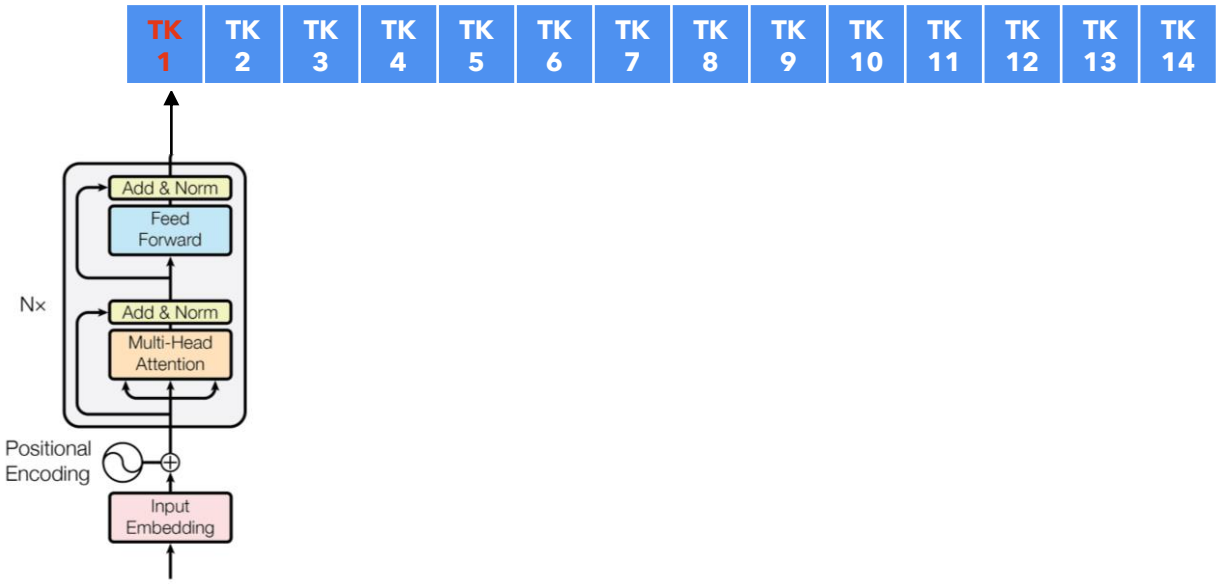
Umar Jamil – https://github.com/hkproj/bert-from-scratch

Text Classification: training

Target (1 token):



Output (16 tokens):



Input (16 tokens): [CLS] My router's led is not working, I tried changing the power socket but still nothing.

Umar Jamil – https://github.com/hkproj/bert-from-scratch

Outline

- Language Models
  - Training
  - Inference
- Transformer architecture (Encoder)
  - Embedding vectors
  - Positional encoding
  - Self attention and causal mask
- BERT
  - The importance of the left and the right context
  - BERT pre-training
    - Masked Language Model task
    - Next Sentence Prediction task
  - BERT fine-tuning
    - Text Classification Task
    - Question Answering Task

Question Answering

Question answering is the task of answering questions given a context.

Context: “Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.”

Question: "What is the fashion capital of China?"

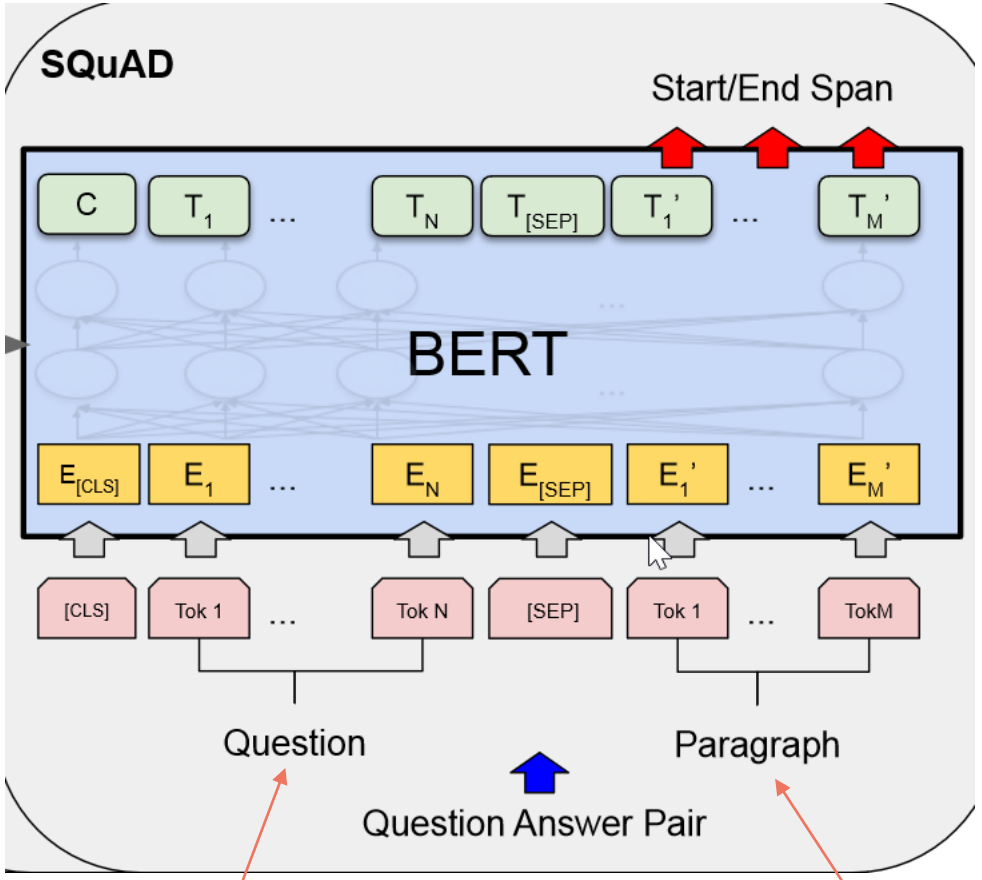
Answer: “Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.”

The model has to highlight the part of text that contains the answer.

Two problems:

1. We need to find a way for BERT to understand which part of the input is the context, which one is the question.
2. We also need to find a way for BERT to tell us where the answer starts and where it ends in the context provided.

Question Answering: sentence A and B



Umar Jamil – https://github.com/hkproj/bert-from-scratch

Question Answering: **start** and **end** positions

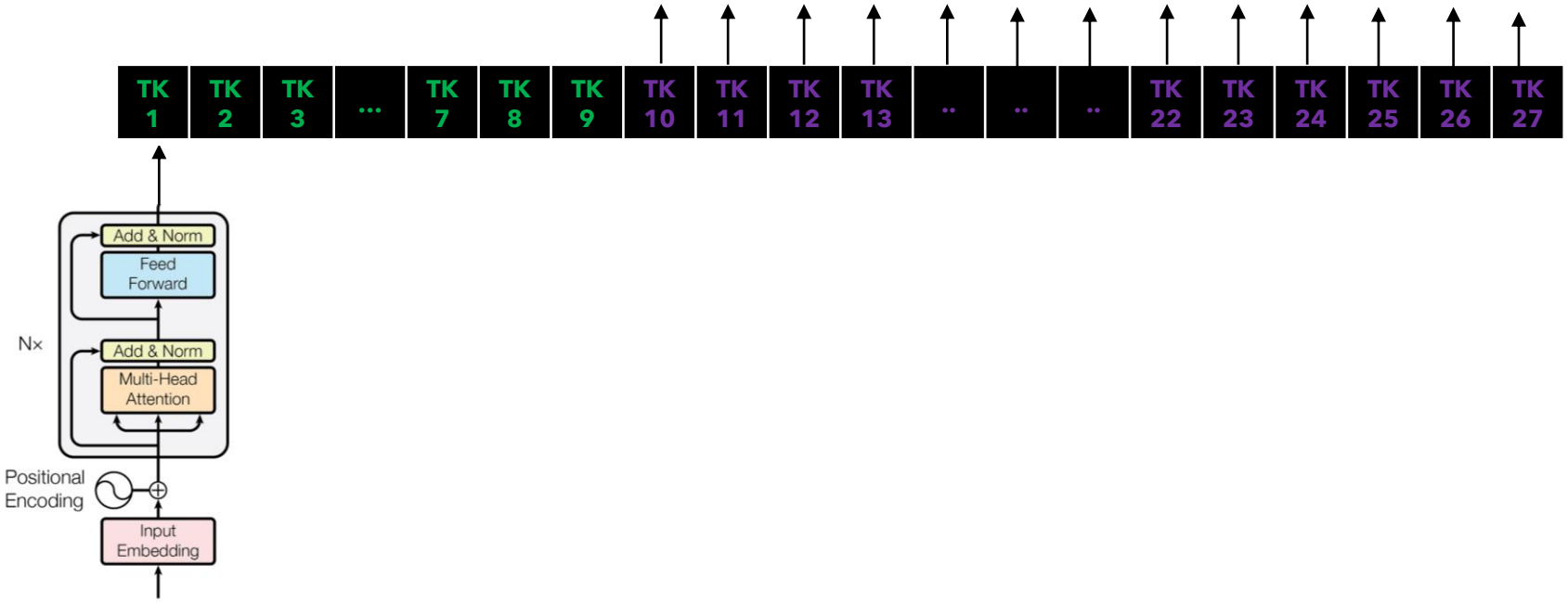
Target (1 token):

start=TK10, end=TK10

Loss → Run **backpropagation** to update the weights

Linear Layer (2 output features) + **Softmax**

Output (27 tokens):



Input (27 tokens):

[CLS] What is the fashion capital of China? [SEP] Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city.

Thanks for watching!
Don’t forget to subscribe for more amazing content on AI and Machine Learning!