ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 19. Into Transformers[1]

Alberto Barrón-Cedeño
a.barron@unibo.it

28/11/2024

[1] Partially based on medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

Table of Contents

Sequence to Sequence Models

Seq2Seq

Seq2Seq models transform a sequence of elements (e.g., the words in a sentence) into another sequence

Seq2Seq

Seq2Seq models transform *a sequence of elements (e.g., the words in a sentence) into another sequence*

Examples of problems that fit Seq2Seq?

Seq2Seq

Seq2Seq models transform a sequence of elements (e.g., the words in a sentence) into another sequence

Examples of problems that fit Seq2Seq?

- Text simplification

Seq2Seq

Seq2Seq models transform a sequence of elements (e.g., the words in a sentence) into another sequence

Examples of problems that fit Seq2Seq?

- Text simplification
- Paraphrasing

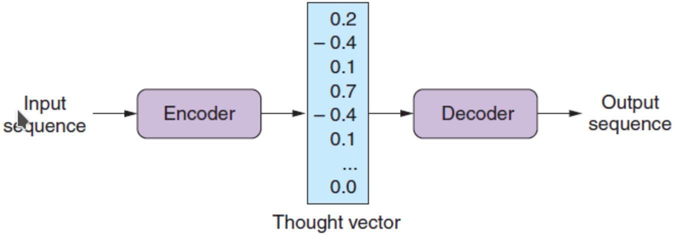
Seq2Seq

> *Seq2Seq models transform a sequence of elements (e.g., the words in a sentence) into another sequence*

Examples of problems that fit Seq2Seq?

- Text simplification
- Paraphrasing
- Machine translation

Seq2Seq
Encoder-Decoder architecture

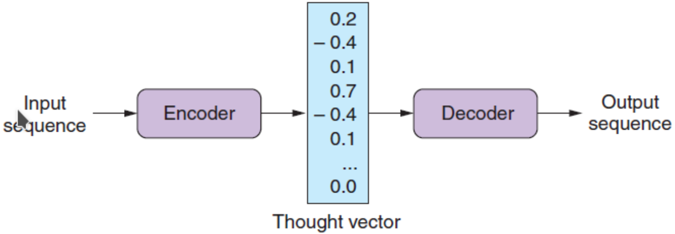


Encoder takes the input sequence and maps it into a higher-dimensional space (vector)

Decoder turns the vector into an output sequence (language, symbols, copy of the input[2])

[2]Smaller vector for compression

(Lane et al., 2019, 315)



Encoder takes the input sequence and maps it into a higher-dimensional space (vector)

Decoder turns the vector into an output sequence (language, symbols, copy of the input[2])

[2]Smaller vector for compression

(Lane et al., 2019, 315)

Seq2Seq
Intuition[3]

- I need to translate texts from Italian to English

[3] From medium

Seq2Seq

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob

[3] From medium

Seq2Seq

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian

[3] From medium

Seq2Seq

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian
  - Both speak (just a bit of!) Spanish

[3] From medium

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian
  - Both speak (just a bit of!) Spanish

What do I need to get Alice and Bob to translate properly together?

[3] From medium

Seq2Seq

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian
  - Both speak (just a bit of!) Spanish

What do I need to get Alice and Bob to translate properly together?

I need to teach them better Spanish

[3] From medium

Seq2Seq
Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian
  - Both speak (just a bit of!) Spanish

What do I need to get Alice and Bob to translate properly together?
I need to teach them better Spanish

Alice	is my encoder
Spanish	is the *language* of my thought vector
Bob	is my decoder

[3] From medium

Seq2Seq

Intuition[3]

- I need to translate texts from Italian to English
- I have two *translators*: Alice and Bob
  - Alice speaks Italian, but not English
  - Bob Speaks English, but not Italian
  - Both speak (just a bit of!) Spanish

What do I need to get Alice and Bob to translate properly together?
I need to teach them better Spanish

Alice is my encoder

Spanish is the *language* of my thought vector

Bob is my decoder

I need to learn (train) the model to encode/decode the text

[3] From medium

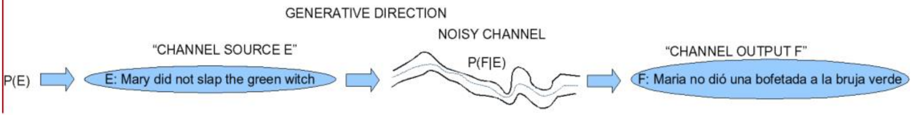


Diagram from Jurafsky's `https://image1.slideserve.com/1844322/the-noisy-channel-model-for-mt-l.jpg`

Attention is all you need

Attention (Vaswani et al., 2017)

The attention-mechanism looks at an input sequence and decides, at each step, which other parts of the sequence are important[4]

[4]Memory in an LSTM rings a bell?

Attention (Vaswani et al., 2017)

The attention-mechanism looks at an input sequence and decides, at each step, which other parts of the sequence are important[4]

Encoder (LSTM) uses the attention mechanism to take into account several other inputs for each element in the input

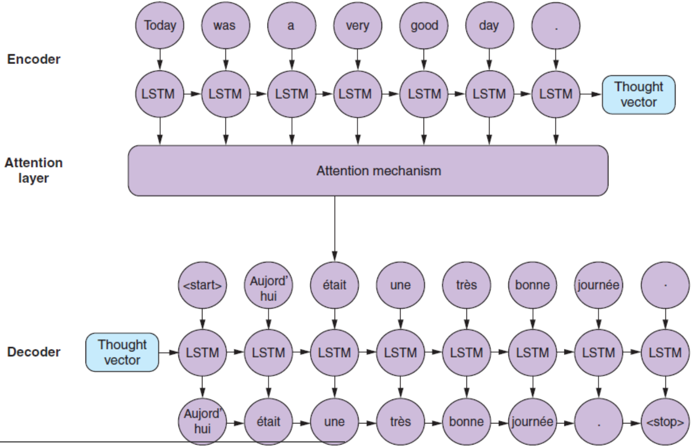
Decoder (LSTM) takes both the encoded sentence and the weights from the attention mechanism.

[4] Memory in an LSTM rings a bell?

Attention

Sequence Labelling

- Part-of-speech tagging
- Dependency parsing
- Named entity recognition



(Lane et al., 2019, 334)

Transformers

Transformers

A Transformer [. . .] helps in transforming one sequence of input into another depending on the problem statement. Examples:

- Translation from one language to another
- Paraphrasing
- Question answering

---

https://medium.com/data-science-in-your-pocket/attention-is-all-you-need-understanding-with-example-c8d074c37767

Transformers

A Transformer [. . .] helps in transforming one sequence of input into another depending on the problem statement. Examples:

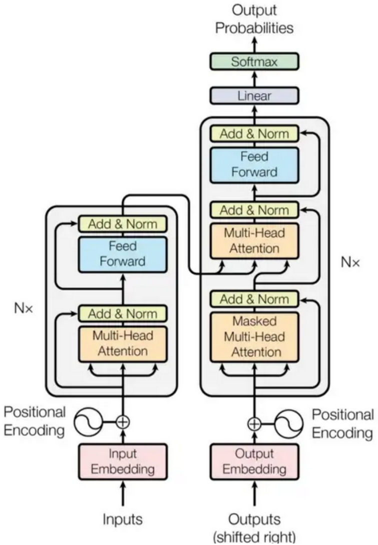
- Translation from one language to another
- Paraphrasing
- Question answering

No recurrent neural networks in this case

https://medium.com/data-science-in-your-pocket/attention-is-all-you-need-understanding-with-example-c8d074c37767

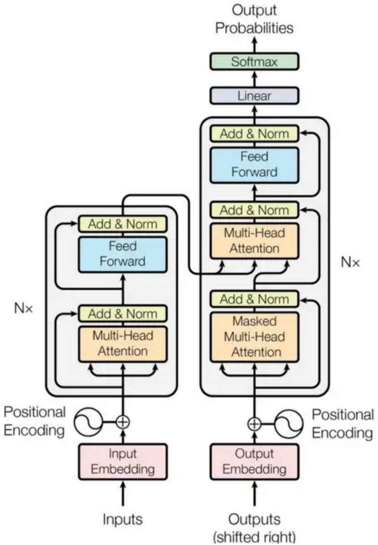
Transformers

Architecture (Vaswani et al., 2017)



Transformers

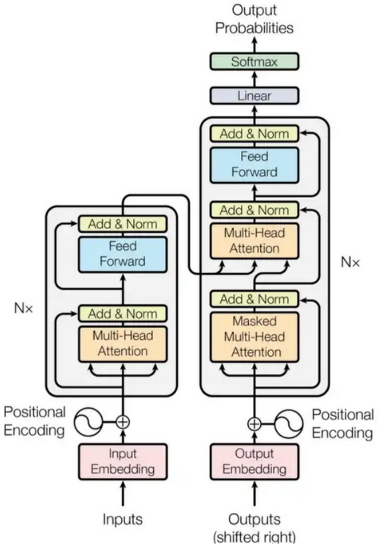
Architecture (Vaswani et al., 2017)



- Encoder on the left, Decoder on the right

Transformers

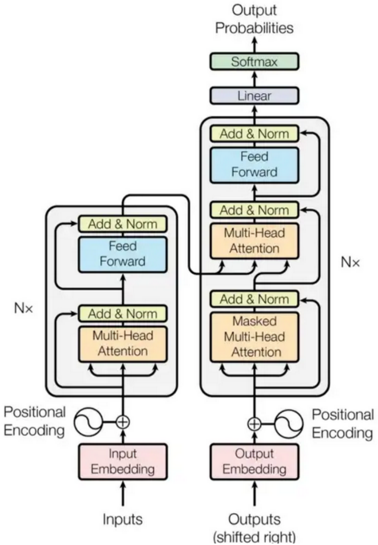
Architecture (Vaswani et al., 2017)



- Encoder on the left, Decoder on the right
- Both can be stacked on top of each other multiple times: Nx (=6)

Transformers

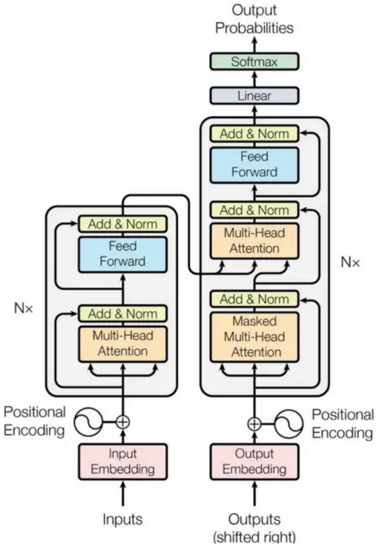
Architecture (Vaswani et al., 2017)



- Encoder on the left, Decoder on the right
- Both can be stacked on top of each other multiple times: Nx (=6)
- Prominent layers
  - Multi-Head Attention
  - Feed-forward

Transformers

Architecture (Vaswani et al., 2017)

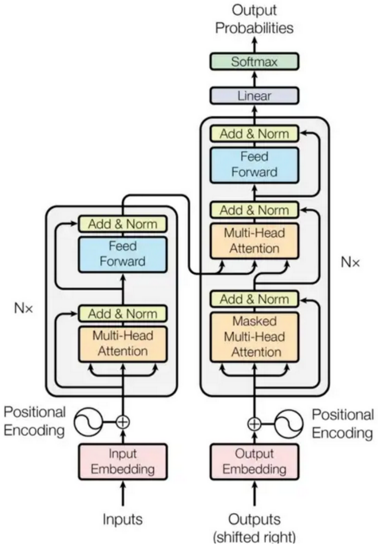


- Encoder on the left, Decoder on the right
- Both can be stacked on top of each other multiple times: Nx (=6)
- Prominent layers
  - Multi-Head Attention
  - Feed-forward

- Embedding: input/output are embedded into an n-dimensional space

Transformers

Architecture (Vaswani et al., 2017)



- Encoder on the left, Decoder on the right
- Both can be stacked on top of each other multiple times: Nx (=6)
- Prominent layers
  - Multi-Head Attention
  - Feed-forward

- Embedding: input/output are embedded into an n-dimensional space
- Positional encoding: gives the relative position of each word in the input/output[a]

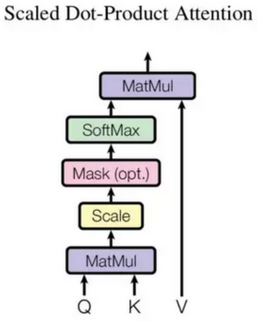
[a] This is not a recurrent network

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key

(Vaswani et al., 2017)

Transformers

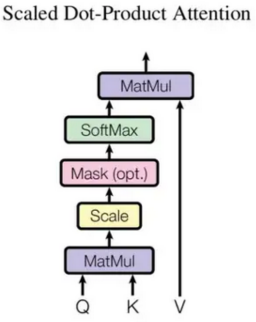
Muli-head attention (Vaswani et al., 2017)



Transformers

Muli-head attention (Vaswani et al., 2017)

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K_T}{\sqrt{d_k}}\right) V$$

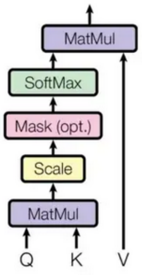


Transformers

Muli-head attention (Vaswani et al., 2017)

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K_T}{\sqrt{d_k}}\right) V$$

Scaled Dot-Product Attention

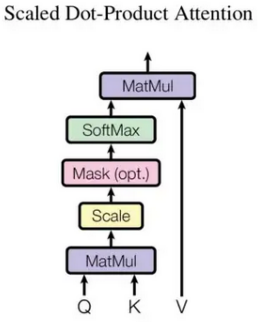


- Q queries: vector representation of one word in the sequence
- K keys: to the vector representations for all the words in the sequence
- V values of the vector representations for all the words in the sequence (same as Q)[a]
- d_k Dimension of Q and K

Transformers

Muli-head attention (Vaswani et al., 2017)

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K_T}{\sqrt{d_k}}\right) V$$



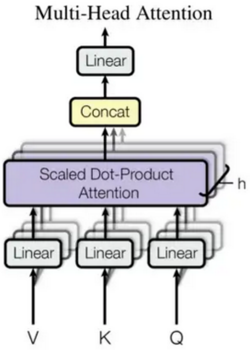
- Q queries: vector representation of one word in the sequence
- K keys: to the vector representations for all the words in the sequence
- V values of the vector representations for all the words in the sequence (same as Q)[a]
- d_k Dimension of Q and K

$Attention(Q, K, C)$ weights on the values

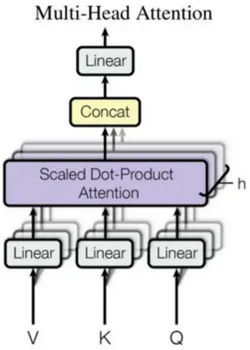
[a] There is a trick here: actually, we have $q \in Q \forall q$ in the

Transformers

Muli-head attention (Vaswani et al., 2017)



"Linearly project[ing] the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions

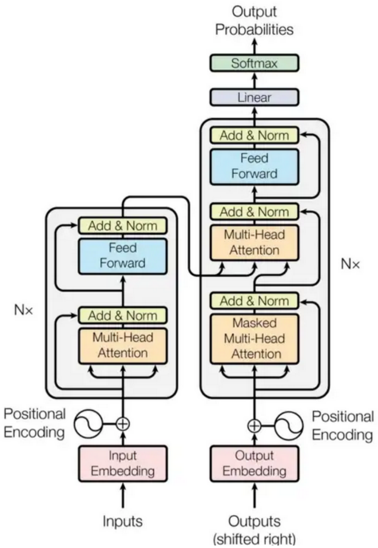


"Linearly project[ing] the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions

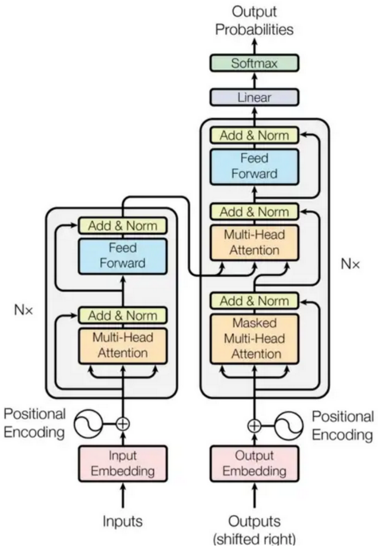
Matrices W that are learned (rings a bell?)

Transformers

Attention in words



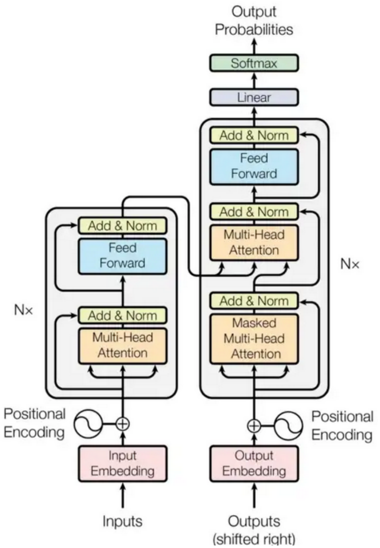
- The weights define how each word in sequence Q is influenced by all other words in the sequence (K)



- The weights define how each word in sequence Q is influenced by all other words in the sequence (K)
- SoftMax distributes the weight over all words $(\sum_K = 1)$

Transformers

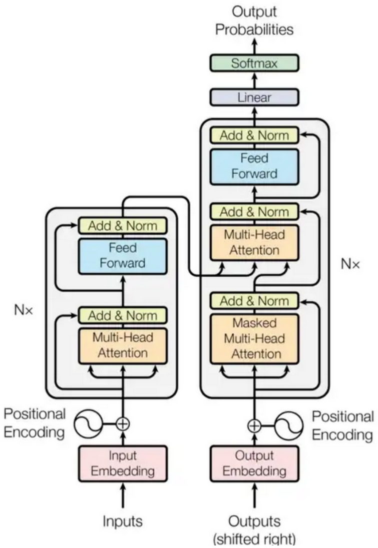
Attention in words



- The weights define how each word in sequence Q is influenced by all other words in the sequence (K)
- SoftMax distributes the weight over all words $(\sum_K = 1)$
- The weights are applied to all the words in sequence V

Transformers

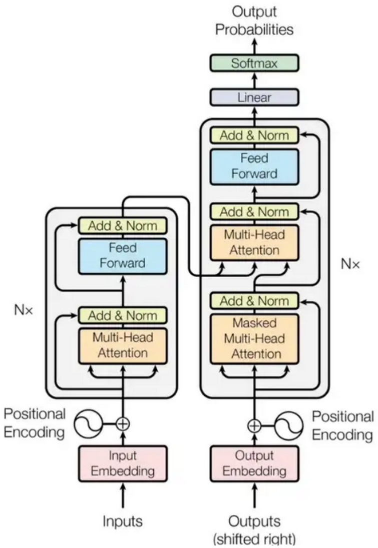
Attention in words



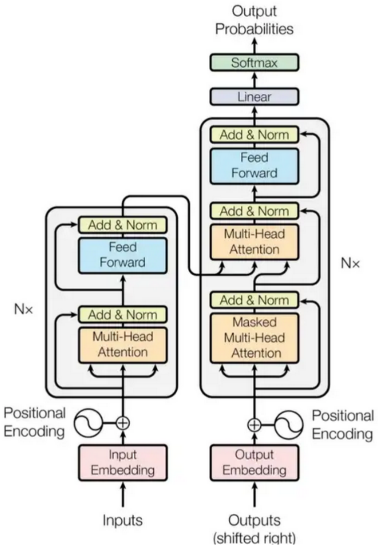
- The weights define how each word in sequence Q is influenced by all other words in the sequence (K)
- SoftMax distributes the weight over all words ($\sum_K = 1$)
- The weights are applied to all the words in sequence V
- Matrices Q, K, and V are different for each attention module

Transformers

Attention in words



- The weights define how each word in sequence Q is influenced by all other words in the sequence (K)
- SoftMax distributes the weight over all words ($\sum_K = 1$)
- The weights are applied to all the words in sequence V
- Matrices Q, K, and V are different for each attention module
- The module connecting encoder and decoder takes into account the encoder input-sequence together with the decoder input-sequence up to a given position

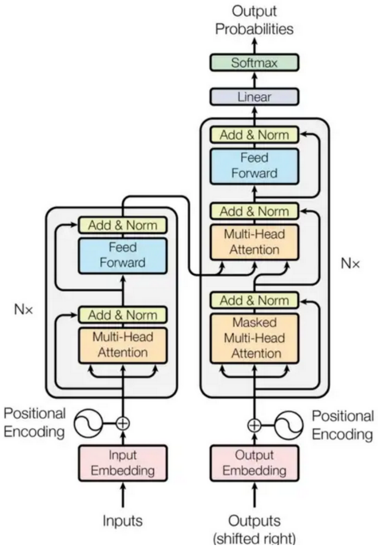


input$_e$ x_0 x_1 x_2 x_3 $\ldots$ $x_{|X|}$

input$_d$ y_1 y_2 y_3 y_4 $\ldots$ $y_{|Y+1|}$

Transformers

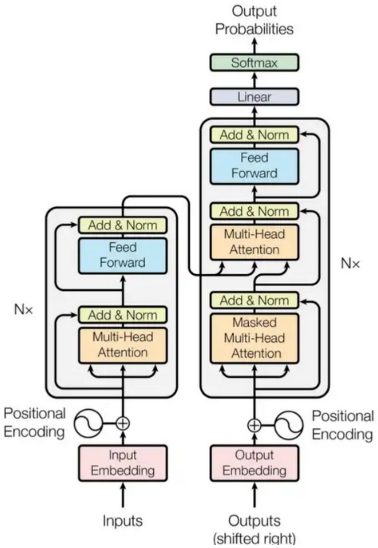
Training



input$_e$ x_0 x_1 x_2 x_3 $\dots$ $x_{|X|}$

input$_d$ y_1 y_2 y_3 y_4 $\dots$ $y_{|Y+1|}$

Why shifting input$_d$?

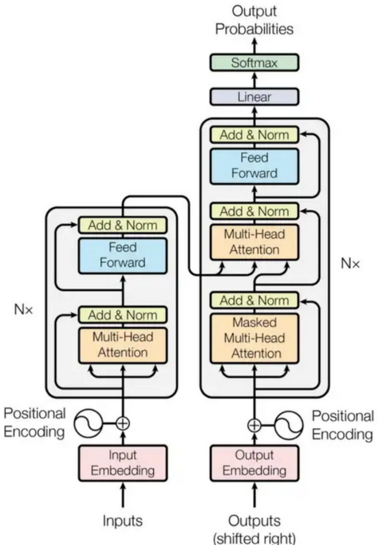


input$_e$ $x_0\ x_1\ x_2\ x_3\ \dots\ x_{|X|}$

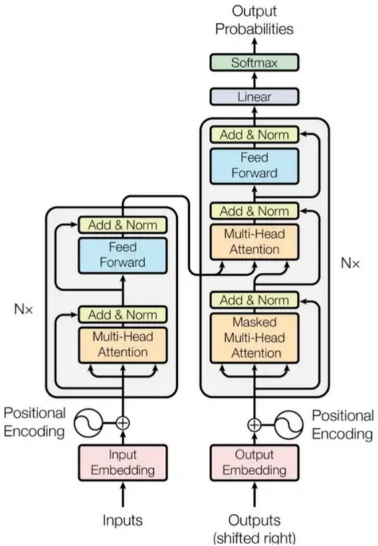
input$_d$ $y_1\ y_2\ y_3\ y_4\ \dots\ y_{|Y+1|}$

Why shifting input$_d$?

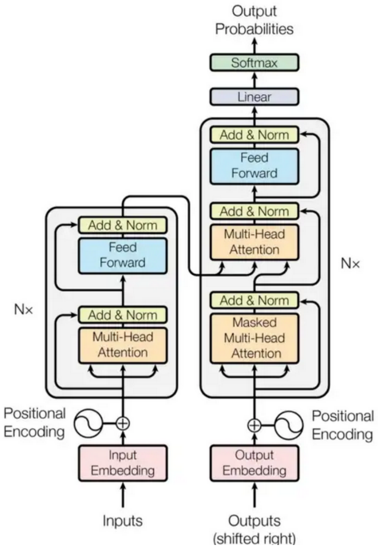
We want to learn that, given the encoder sequence and a particular decoder sequence (both seen already by the model), we have to predict the next word/character (otherwise, the model learns to copy the input$_d$)



- Input the full input$_e$ and an empty input$_d$ (start-of-sentence token)

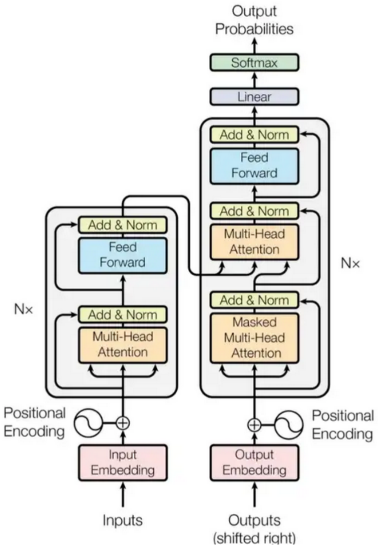


- Input the full input$_e$ and an empty input$_d$ (start-of-sentence token)
- Get the first element of the output produced



- Input the full input$_e$ and an empty input$_d$ (start-of-sentence token)
- Get the first element of the output produced
- Input the full input$_e$ and start-of-sentence + first output element

Transformers
Inference



- Input the full input$_e$ and an empty input$_d$ (start-of-sentence token)
- Get the first element of the output produced
- Input the full input$_e$ and start-of-sentence + first output element
- Repeat until end-of-sentence

References I

Lane, H., C. Howard, and H. Hapkem
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning Publication Co.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin
2017. Attention is all you need.