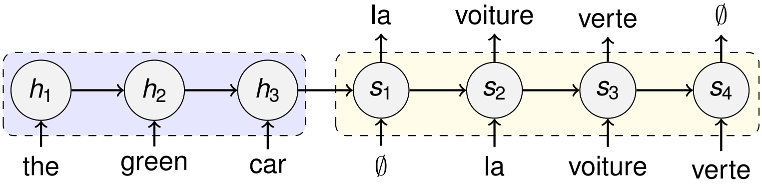
DD2417
9a: The Transformer architecture

Johan Boye, KTH

Alignment/Attention mechanism



	la	voiture	verte
the	0.88	0.01	0.01
green	0.01	0.01	0.89
car	0.11	0.98	0.10

Dependencies in text

Tom visited his brother although he didn't like him much.

Dependencies in text

Tom visited his brother although he didn't like him much.

Although he was sick, Tom went to work.

Self-attention

A spring broke in the bicycle. 

He went out in the spring sun. 

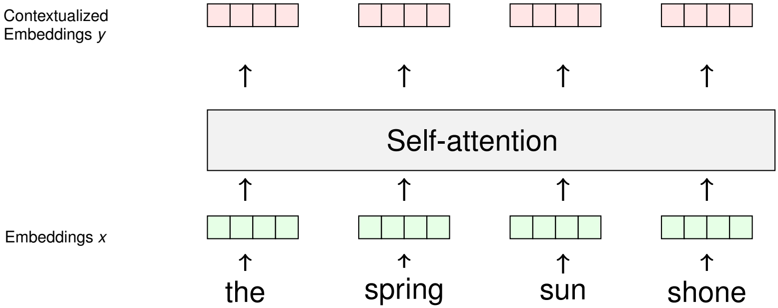
Transformers

Vaswani et al (2017) presented an alternative translation scheme, the *Transformer* architecture:

Two key ideas:

- Self-attention: Producing contexualized representations of the input words.
- No RNNs: An architecture that allows for parallelization of the training

Contextualized representations



Self-attention

	the	spring	sun	shone
the	0.20	0.20	0.55	0.05
spring	0.10	0.40	0.3	0.20
sun	0.10	0.15	0.35	0.40
shone	0.10	0.20	0.40	0.30

Self-attention

	the	spring	sun	shone	SUM
the	0.20	0.20	0.55	0.05	1
spring	0.10	0.40	0.3	0.20	1
sun	0.10	0.15	0.35	0.40	1
shone	0.10	0.20	0.40	0.30	1

Self-attention

	the	spring	sun	shone
the	0.20	0.20	0.55	0.05
spring	0.10	0.40	$\boxed{0.3}$	0.20
sun	0.10	0.15	0.35	0.40
shone	0.10	0.20	0.40	0.30

How much of “sun” do we want to
incorporate into the representation of “spring”?

Self-attention

	x_1	x_2	x_3	x_4
x_1	α_{11}	α_{12}	α_{13}	α_{14}
x_2	α_{21}	α_{22}	$\boxed{\alpha_{23}}$	α_{24}
x_3	α_{31}	α_{32}	α_{33}	α_{34}
x_4	α_{41}	α_{42}	α_{43}	α_{44}

How much of x_3 do we want to incorporate into the representation of x_2?

Self-attention

$$
\begin{aligned}
y_1 &= \alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + \alpha_{14}x_4 \\
y_2 &= \alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}x_3 + \alpha_{24}x_4 \\
y_3 &= \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_{33}x_3 + \alpha_{34}x_4 \\
y_4 &= \alpha_{41}x_1 + \alpha_{42}x_2 + \alpha_{43}x_3 + \alpha_{44}x_4
\end{aligned}
$$

$$\Uparrow$$

	x_1	x_2	x_3	x_4
x_1	α_{11}	α_{12}	α_{13}	α_{14}
x_2	α_{21}	α_{22}	α_{23}	α_{24}
x_3	α_{31}	α_{32}	α_{33}	α_{34}
x_4	α_{41}	α_{42}	α_{43}	α_{44}

Self-attention

$$y_1 \; y_2 \; y_3 \; y_4$$

$$\Uparrow$$

$$\begin{aligned}
y_1 &= \alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + \alpha_{14}x_4 \\
y_2 &= \alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}x_3 + \alpha_{24}x_4 \\
y_3 &= \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_{33}x_3 + \alpha_{34}x_4 \\
y_4 &= \alpha_{41}x_1 + \alpha_{42}x_2 + \alpha_{43}x_3 + \alpha_{44}x_4
\end{aligned}$$

$$\Uparrow$$

	x_1	x_2	x_3	x_4
x_1	α_{11}	α_{12}	α_{13}	α_{14}
x_2	α_{21}	α_{22}	α_{23}	α_{24}
x_3	α_{31}	α_{32}	α_{33}	α_{34}
x_4	α_{41}	α_{42}	α_{43}	α_{44}

$$\Uparrow$$

$$x_1 \; x_2 \; x_3 \; x_4$$

$\Leftarrow$ Self-attention (first attempt)

Self-attention: scores

We want to compute a score expressing relevant x_i is to x_j.

Idea from w2v and Glove: Score is computed by the dot product:

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$

The *higher* the score, the *more relevant* x_j is to x_i

- and the more we want to incorporate x_j into y_i, the contextualized representation of x_i

However: dot product is symmetric, but x_j might be more relevant to x_i than the other way around.

Queries, keys, and values

Words are being used in three roles: As *queries*, *keys*, and *values*.

$$
\begin{aligned}
y_1 &= \alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + \alpha_{14}x_4 \\
y_2 &= \alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}\boxed{x_3} + \alpha_{24}x_4 \\
y_3 &= \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_{33}x_3 + \alpha_{34}x_4 \\
y_4 &= \alpha_{41}x_1 + \alpha_{42}x_2 + \alpha_{43}x_3 + \alpha_{44}x_4
\end{aligned}
$$

$$\Uparrow$$

$$
\begin{array}{c|cccc}
 & x_1 & \boxed{x_2} & x_3 & x_4 \\
\hline
x_1 & \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\
\boxed{x_2} & \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\
x_3 & \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\
x_4 & \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44}
\end{array}
$$

Value

Key

Query

Queries, keys, and values

A word vector x_i is being used in three different roles:

- as an input (a key)
- as the focus of attention when compared to all inputs $x_1 \ldots x_n$ (a query)
- when computing the contextualized output vectors (a value)

Therefore, from each input vector x_i we will produce three vectors:

- a key vector k_i
- a query vector q_i
- a value vector v_i

Queries, keys, and values

We define three trainable matrices W^Q, W^K, W^V such that:

- W^Q, W^K have dimensions $d \times d_k$, where d is the dimensionality of the input vectors x_i
- W^V has dimensions $d \times d$
- $q_i = W^Q x_i, \quad k_i = W^K x_i, \quad v_i = W^V x_i$

We can now redefine the score function into: $\text{score}(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$

(we divide by $\sqrt{d_k}$ to avoid getting too big values)

and compute the proportions using softmax: $\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_k \exp(\text{score}(x_i, x_k))}$

Queries, keys, and values

$$
\begin{aligned}
y_1 &= \alpha_{11} v_1 + \alpha_{12} v_2 + \alpha_{13} v_3 + \alpha_{14} v_4 \\
y_2 &= \alpha_{21} v_1 + \alpha_{22} v_2 + \alpha_{23} \boxed{v_3} + \alpha_{24} v_4 \\
y_3 &= \alpha_{31} v_1 + \alpha_{32} v_2 + \alpha_{33} v_3 + \alpha_{34} v_4 \\
y_4 &= \alpha_{41} v_1 + \alpha_{42} v_2 + \alpha_{43} v_3 + \alpha_{44} v_4
\end{aligned}
$$

$$\Uparrow$$

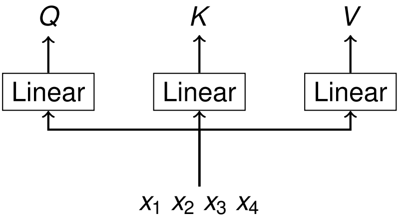
	k_1	$\boxed{k_2}$	k_3	k_4
q_1	α_{11}	α_{12}	α_{13}	α_{14}
$\boxed{q_2}$	α_{21}	α_{22}	α_{23}	α_{24}
q_3	α_{31}	α_{32}	α_{33}	α_{34}
q_4	α_{41}	α_{42}	α_{43}	α_{44}

Value

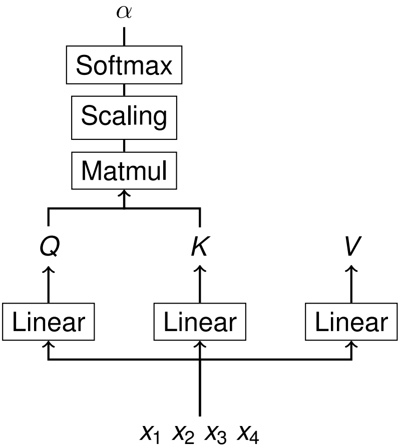
Key

Query

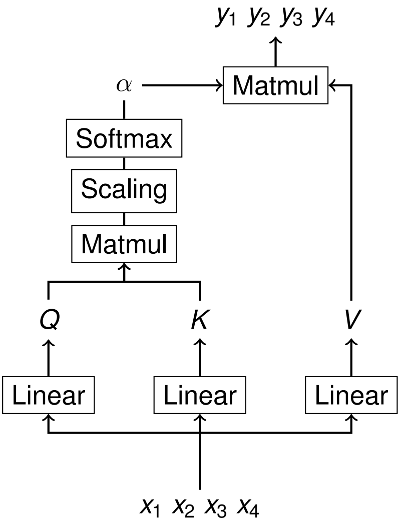
Self-attention



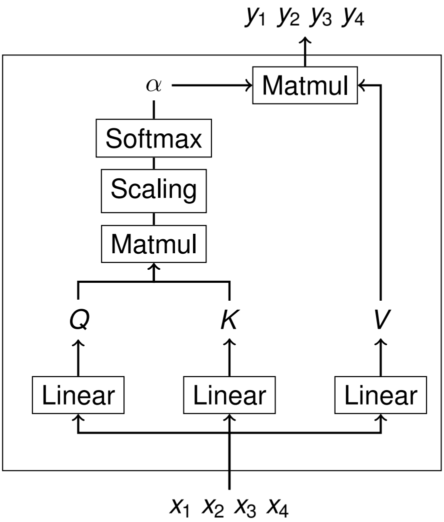
Self-attention



Self-attention



Self-attention



$\Leftarrow$ Self-attention (second attempt)

Self-attention

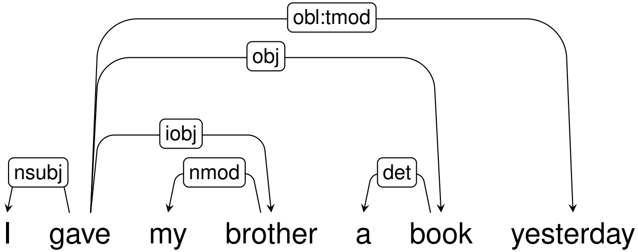
As an equation, self-attention can be expressed:

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

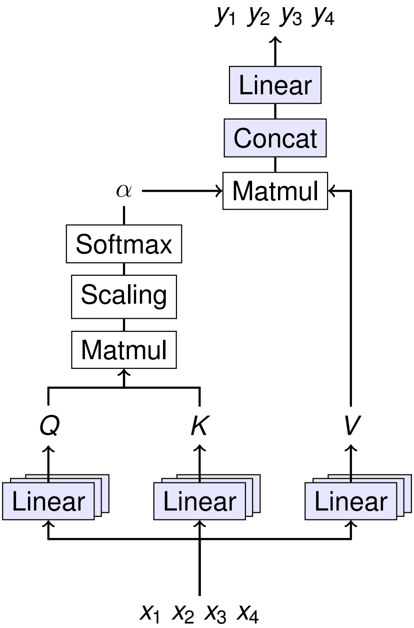
Vaswani et al. (2017) "Attention is all you need"

Multi-head self-attention

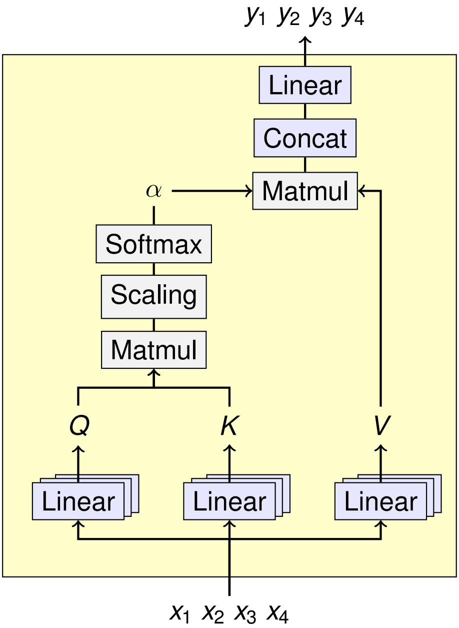
A word can need to “pay attention” to many other words, for different reasons.



Multi-head self-attention

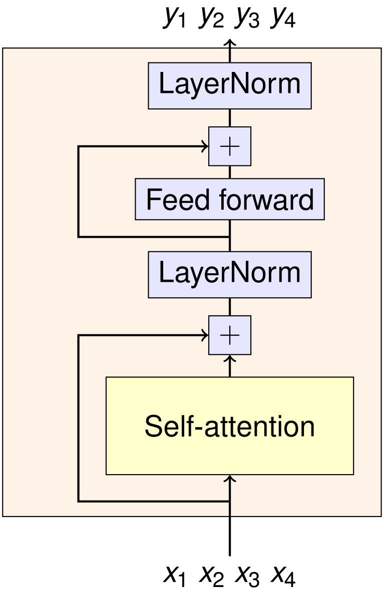


Multi-head self-attention

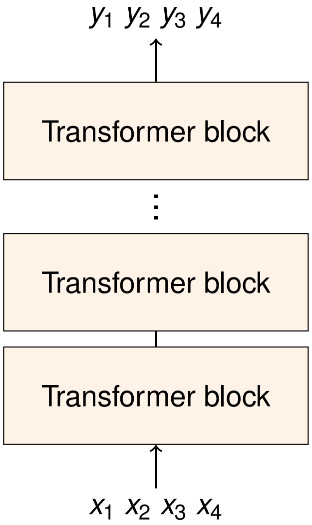


$\Leftarrow$ Self-attention (third attempt)

Transformer block



Transformer encoder



Word order

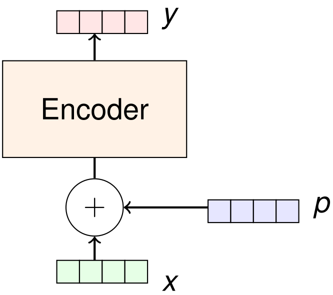
	the	spring	sun	shone
the	0.20	0.20	0.55	0.05
spring	0.10	0.40	$\boxed{0.3}$	0.20
sun	0.10	0.15	0.35	0.40
shone	0.10	0.20	0.40	0.30

Word order is not represented in this table, but word order matters!

“Mary is smarter than Tom.”
“Tom is smarter than Mary.”

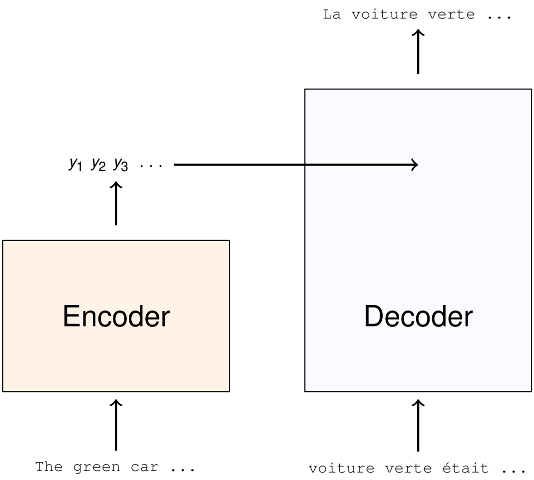
Positional encoding

To inject word-order information, each input vector x_i is added to a *positional vector* p_i before applying the transformer encoder (or decoder).



Positional vectors could either be *learned* or *generated by a function*.

Original transformer architecture



DD2417
9b: Masked language models and BERT

Johan Boye, KTH

BERT

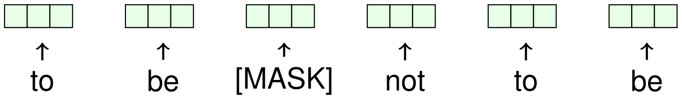
- BERT – Bidirectional Encoder Representations from Transformers
- A general language model based on the Transformer encoder
- Pre-trained on two objectives:
  - Unmasking words
  - Next sentence prediction
- Through *finetuning* BERT, one can solve a number of language engineering problem.

Unmasking words

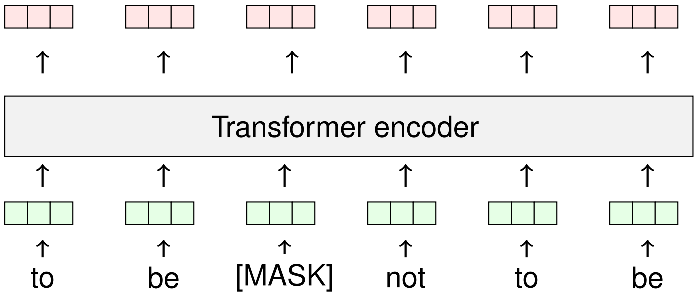
A fill-in-the-gap test:

`To be ________ not to be`

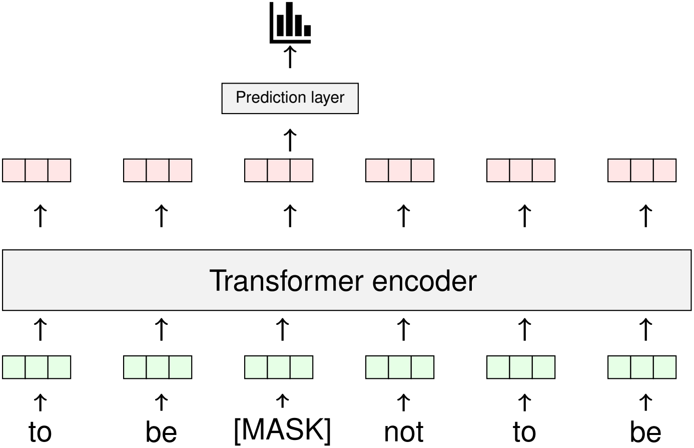
Unmasking words



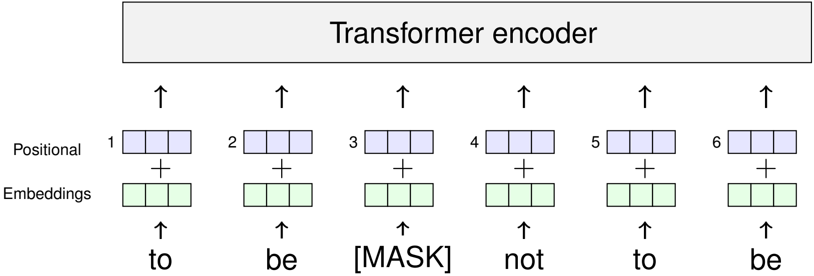
Unmasking words



Unmasking words



Adding positional vectors



Subword tokenization

Treating every unique word as a unique token class has its drawbacks:

- Some words are rare
  - (e.g. *ultracrepedarian*)
- Similar-looking words can mean similar things
  - (e.g. *contraction–contractions*)

Thus, *subword tokenization* makes a lot of sense.

Subword tokenization

ultracrepedarian → `'ultra', '##cre', '##ped', '##arian'`

contractions → `'contraction', '##s'`

debug → `'de', '##bu', '##g'`

BERT uses 30,000 token classes, computed automatically.

The *byte-pair encoding* (BPE) algorithm can be used (ch 2.4.3 in the textbook).

Next sentence prediction

Understanding the logical order of sentences:

Stockholm is the capital of Sweden. One million people live there.

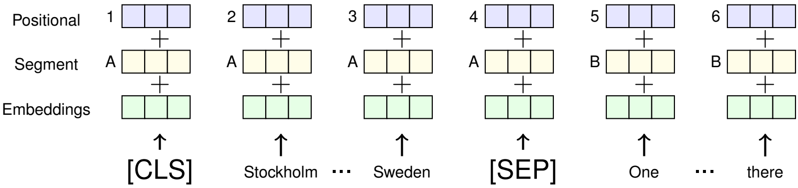
is OK, but not

One million people live there. Stockholm is the capital of Sweden.

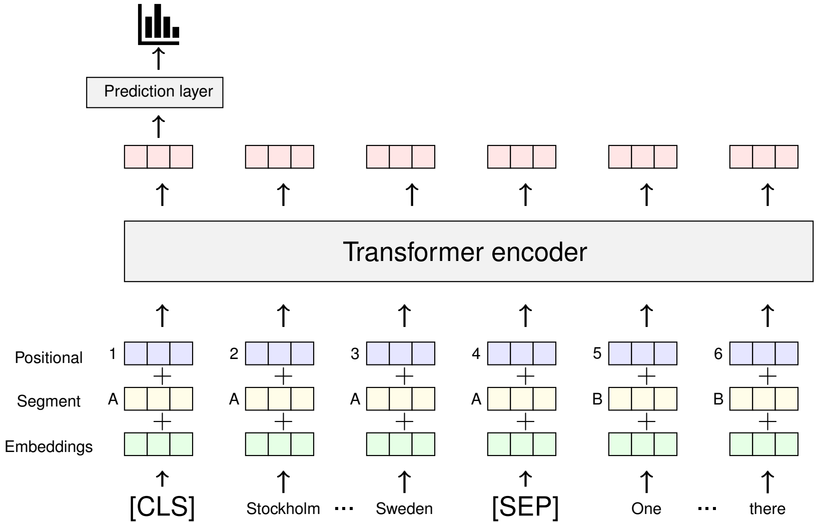
Adding segments

[CLS] Stockholm ··· Sweden [SEP] One ··· there

Adding segments



Adding segments

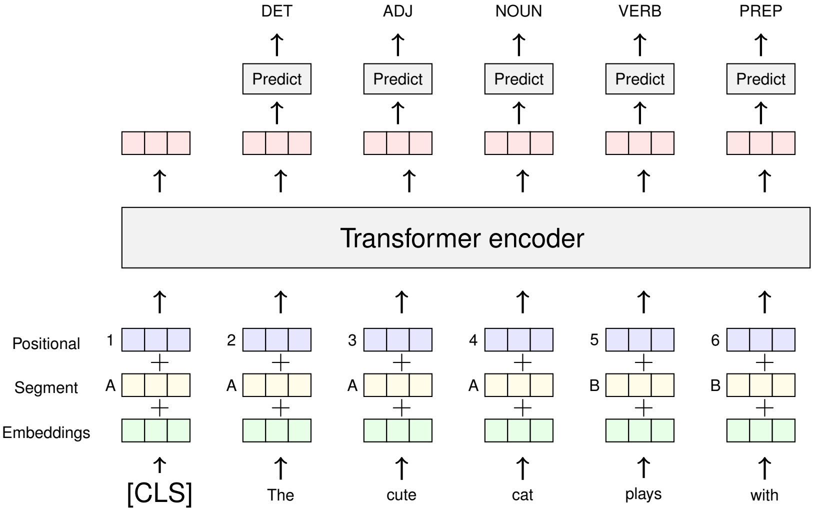


Fine-tuning BERT

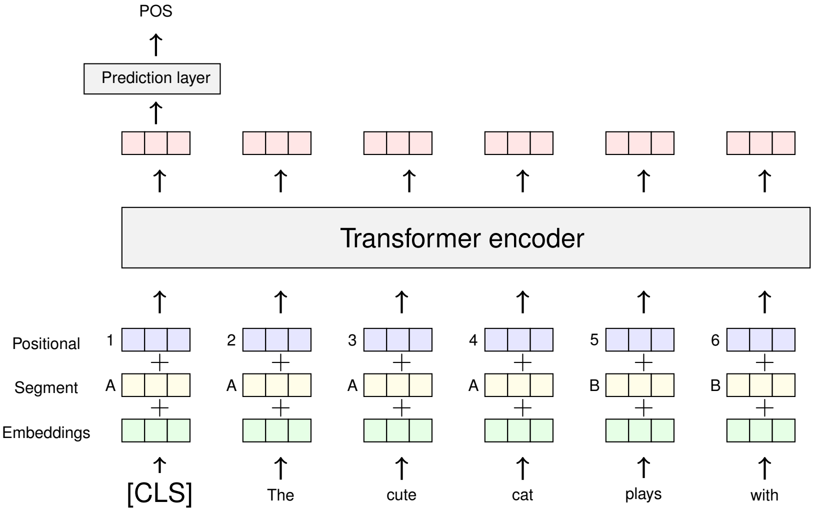
Through *finetuning* BERT, one can solve a number of language engineering problem.

- Token-based prediction (e.g. POS tagging) by adding a prediction layer on top of every output vector
- Sequence prediction (e.g. sentiment analysis) by adding a prediction layer on top of the CLS output vector

Token classification



Sequence classification



Reading comprehension

Machine reading comprehension can be solved with the token classification approach.

Input: `[CLS]` *`Wikipedia text about Stockholm`* `[SEP]` `Which scientific prize is awarded each year in Stockholm?`

Categorize each token as

- `B`, first word of the answer, or
- `I`, word belongs to the answer, or
- `O`, word is not part of the answer

BERT numbers

BERT Base/Bert Large):

- 12/24 stacked transformer blocks
- 12/16 attention heads
- 768/1024 hidden-vector size
- 512 input tokens (400 words)
- 235M/340M trainable parameters

Many larger/improved variants of BERT have been released 2019-

- More words in the input
- More training data
- Multilingual

DD2417
9c: Autoregressive language models and GPT

Johan Boye, KTH

GPT

- GPT – Generative Pre-trained Transformer
- Developed by OpenAI
  - GPT 1–4 (2018–2022)
- Simpler training objective than BERT:
  - Predict the next token given the preceding ones
- GPT can also be fine-tuned to solve language tasks
  - Notably, InstructGPT, which formed the basis for ChatGPT

Predicting the next token

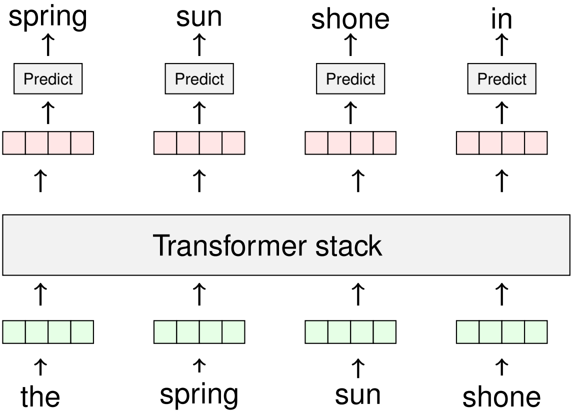
Example: `The cute cat plays with a string`

Training examples:

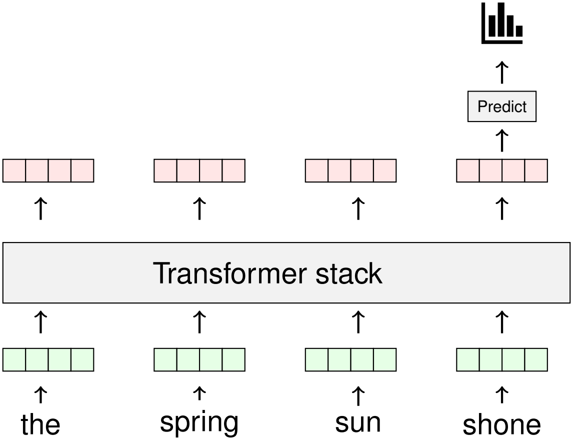
- `cute` follows `The`
- `cat` follows `The cute`
- `plays` follows `The cute cat`
- `with` follows `The cute cat plays`
- *etc.*

Using the Transformer architecture, all these examples can be learnt in parallel.

Training an autoregressive language model



Using an autoregressive language model



Queries, keys, and values

$$y_1 = \alpha_{11} v_1 + \alpha_{12} v_2 + \alpha_{13} v_3 + \alpha_{14} v_4$$
$$y_2 = \alpha_{21} v_1 + \alpha_{22} v_2 + \alpha_{23} \boxed{v_3} + \alpha_{24} v_4$$
$$y_3 = \alpha_{31} v_1 + \alpha_{32} v_2 + \alpha_{33} v_3 + \alpha_{34} v_4$$
$$y_4 = \alpha_{41} v_1 + \alpha_{42} v_2 + \alpha_{43} v_3 + \alpha_{44} v_4$$

$$\Uparrow$$

	k_1	$\boxed{k_2}$	k_3	k_4
q_1	α_{11}	α_{12}	α_{13}	α_{14}
$\boxed{q_2}$	α_{21}	α_{22}	α_{23}	α_{24}
q_3	α_{31}	α_{32}	α_{33}	α_{34}
q_4	α_{41}	α_{42}	α_{43}	α_{44}

Attention for autoregressive models

$$
\begin{aligned}
y_1 &= \alpha_{11} v_1 + \alpha_{12} v_2 + \alpha_{13} v_3 + \alpha_{14} v_4 \\
y_2 &= \alpha_{21} v_1 + \alpha_{22} v_2 + \alpha_{23} v_3 + \alpha_{24} v_4 \\
y_3 &= \alpha_{31} v_1 + \alpha_{32} v_2 + \alpha_{33} v_3 + \alpha_{34} v_4 \\
y_4 &= \alpha_{41} v_1 + \alpha_{42} v_2 + \alpha_{43} v_3 + \alpha_{44} v_4
\end{aligned}
$$

$$\Uparrow$$

	k_1	k_2	k_3	k_4
q_1	α_{11}	$-\infty$	$-\infty$	$-\infty$
q_2	α_{21}	α_{22}	$-\infty$	$-\infty$
q_3	α_{31}	α_{32}	α_{33}	$-\infty$
q_4	α_{41}	α_{42}	α_{43}	α_{44}

GPT-2 numbers

- 12/24/36/48 stacked transformer blocks in GPT2
- 12/16/20/25 attention heads
- 768 hidden-vector size
- 1024 input tokens (800 words)
- 1.5B trainable parameters

It is assumed that...

- GPT-3 has 175B parameters
- GPT-4 has 1000B parameters
- GPT-3 and GPT-4 have a max input size of 4096 tokens (3200 words)