"Attention is all you need".
-My ex.

"The art of knowing is knowing what to ignore."
- Rumi

All images from Wikimedia Commons unless specified.

UNSW SYDNEY

1

Natural Language Processing (NLP)

COMP6713 – 2025 Term 1

Convener

Dr. Aditya Joshi

aditya.joshi@unsw.edu.au

Week 3

Attention! Transformer

Schedule

2025 Term 1

UNSW SYDNEY

2

UNSW SYDNEY | Australia's Global University

Week 3

Attention! Transformer

Attention
Intuition
Attention in recurrent LM & limitations
Attention as key, query, value

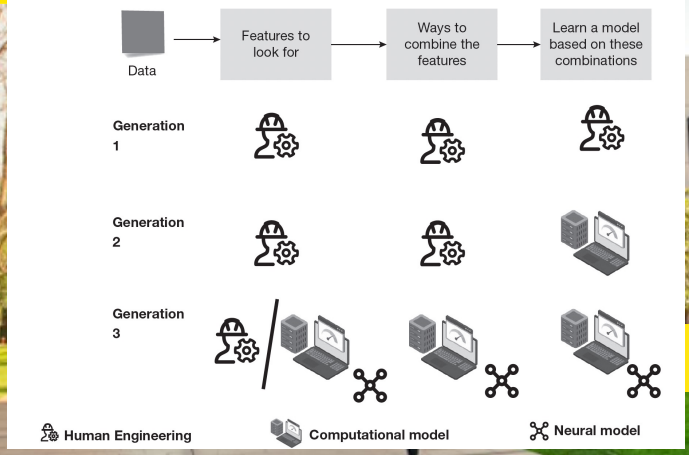
Transformer
Inside-out view of Transformer
Multi-head attention
Encoder
Decoder
Byte-pair encoding
Positional encoding
Impact and legacy of Transformer

And as usual…
code samples, pen-and-paper demos, linguistic examples and dad jokes.

3

UNSW SYDNEY | Australia's Global University

Topics this week



Attention
Transformer

4

Before we begin…

Congratulations on completing the first quiz!

Final exam

No memorization of code syntax: there may be fill in the gaps or 'what does this part of the code do' style questions. Hints along the way in lectures.

Invigilated BYOD exam. Timetable to be communicated.

Individual Assignment:

The individual assignment is now available

Poetry generation ---- a popular use case for neural language models…. but using probabilistic language modeling

Covers material from weeks 1 - 3

Due on Friday of Week 5

Details in your inbox. (Open the assignment document at this point).

UNSW SYDNEY

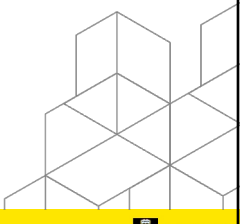
5

..and before we begin…

What do you know about Transformer?

Bullet points are okay.

.. and we will revisit them at the end of week 4.



UNSW SYDNEY

6

Part 0

Why attention?

7

How many cyclists do you see in the picture?

You paid attention to the parts of the **picture** that are important to the key term in the **question.**

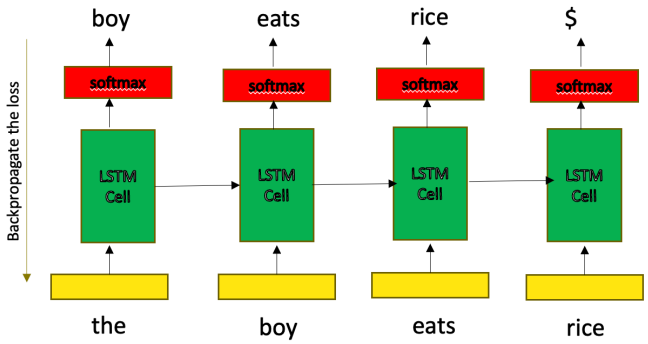
--Cross-attention

Try this: How many piraxes do you see in the picture?

Just kidding.

8

Recap from week 2



Recurrent language models rely on sequential information passing.

Fixed-length vectors become a bottleneck.

9

Recap from week 2

"you can't cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!"

— Ray Mooney (NLP professor at UT Austin)

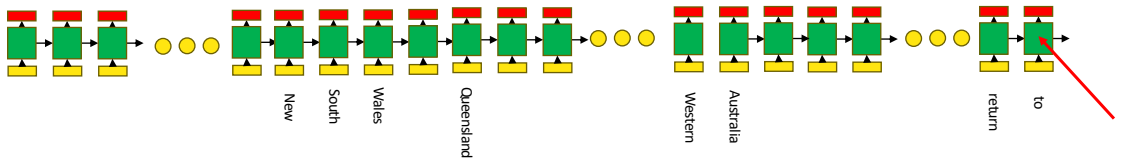
Fixed-length vectors become a bottleneck.

10

Why a bottleneck?

- **Long-range semantics**

Thunderstorms are bringing intense deluges to parts of New South Wales and Queensland, Cyclone Lincoln is likely to re-emerge off the north coast of Western Australia, while elevated fire danger will return to _____



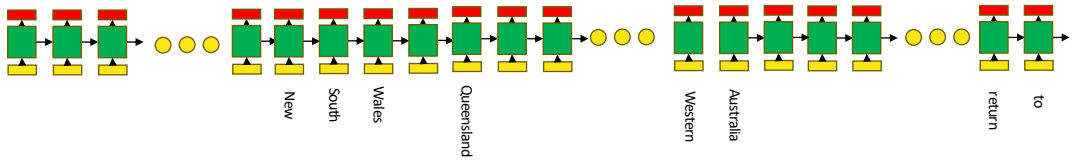
- Coreferential pronoun selection

The restaurant first opened in 1997, and ran successfully until 2023 when ___

Thunderstorm example from: https://www.abc.net.au/news/2024-02-21/thunderstorms-extreme-heat-cyclones-wild-weather-australia/103491052

UNSW SYDNEY

11



When generating the next word in the sequence,

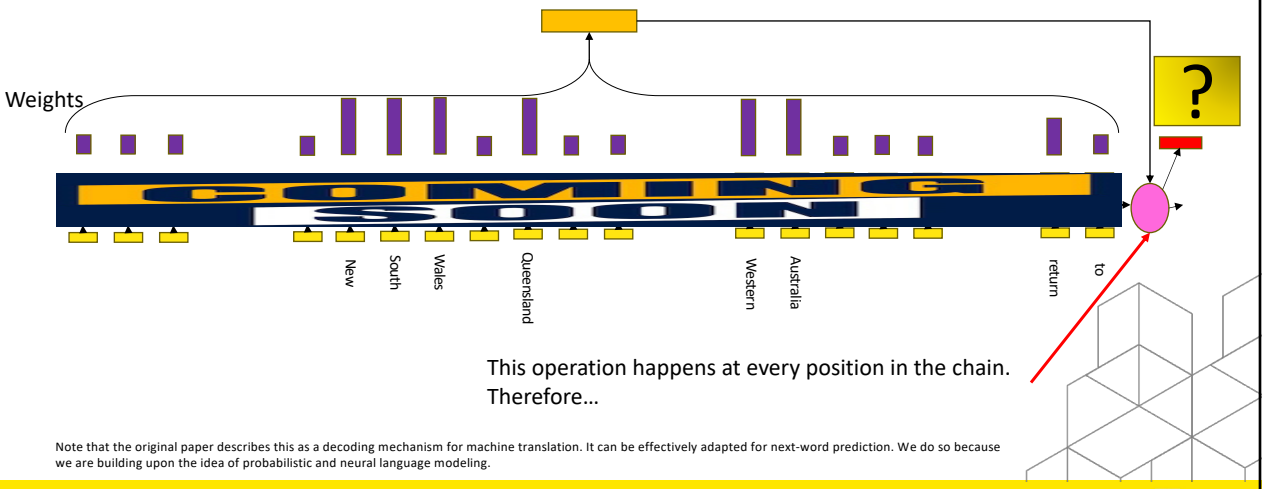
We want to pay 'attention' to some words.

i.e., some words are more important than others.

Note: Bidirectional LSTMs

UNSW SYDNEY

12



This operation happens at every position in the chain.
Therefore…

Note that the original paper describes this as a decoding mechanism for machine translation. It can be effectively adapted for next-word prediction. We do so because we are building upon the idea of probabilistic and neural language modeling.

13

Attention

Attention mechanism is a technique that allows a neural layer to focus on specific parts of a sequence.

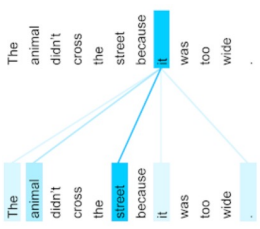
- I missed my bus because it was _____

  Which words are important when predicting the next word in the input?

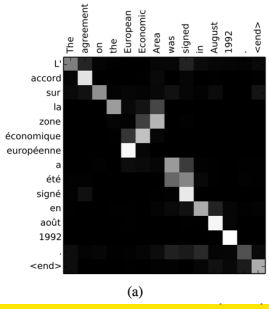
- *Input: Who is the Prime Minister of Australia? Output: _____*

  Which words are important when predicting the next word in the output?

Self-attention models relationships between words in a sentence.



Cross-attention models relationships between words in pairs of sentences.



(a)

https://blog.research.google/2017/08/transformer-novel-neural-network.html
Cross-attention from Bahdanu et al (2015)

14

PyTorch

Originally developed by Meta; now a part of Linux Foundation

Tensor: Similar to numpy arrays, can be manipulated using GPUs

Alternative: TensorFlow

Demo time!

15

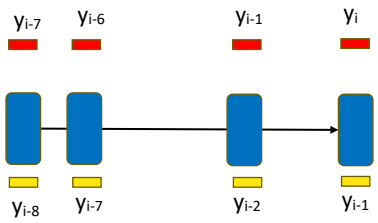
UNSW SYDNEY | Australia's Global University

Part 1

Attention in recurrent language modeling

16

Math board



Advanced Reading (Optional): xLSTM
https://arxiv.org/pdf/2405.04517

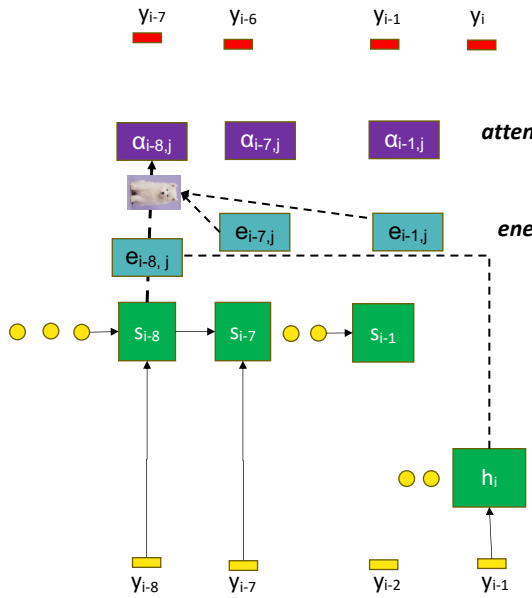
Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate.". ICLR 2015.

17

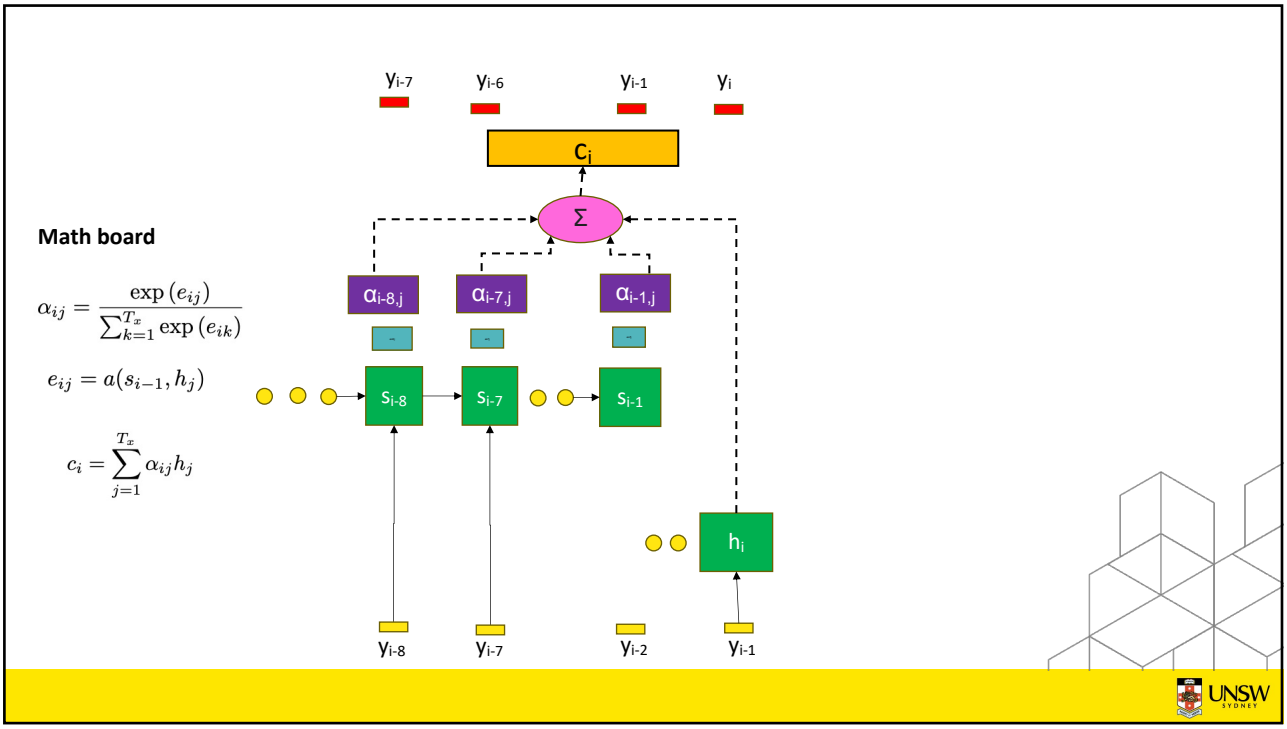
Math board

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

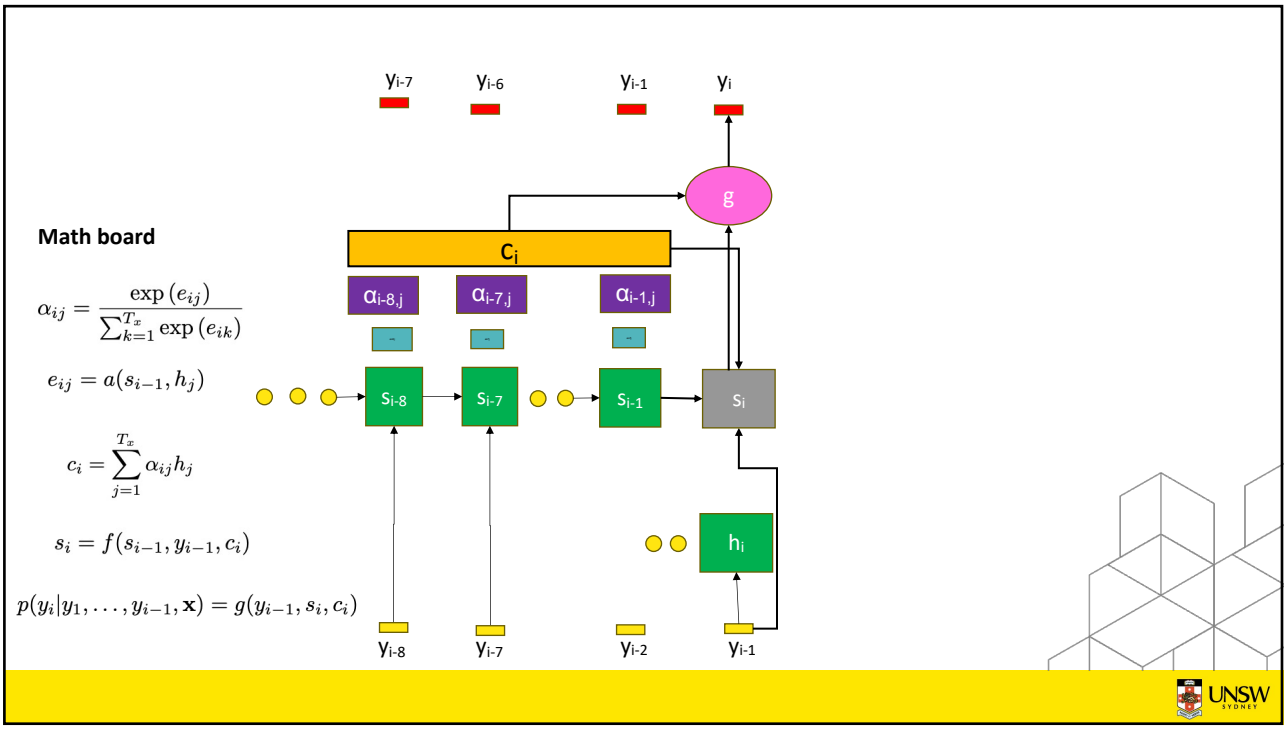
$$e_{ij} = a(s_{i-1}, h_j)$$



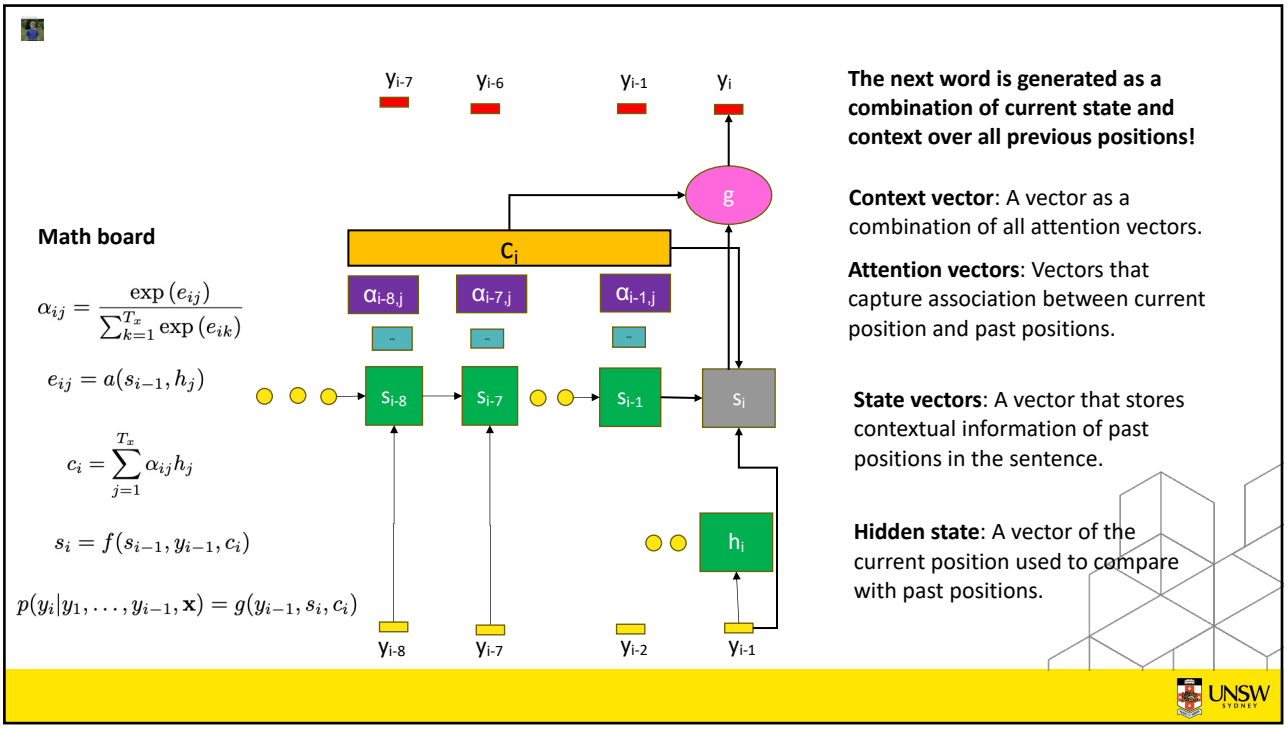
18


Math board
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
$e_{ij} = a(s_{i-1}, h_j)$
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
yi-7
yi-6
yi-1
yi
ci
Σ
αi-8,j
αi-7,j
αi-1,j
si-8
si-7
si-1
hj
yi-8
yi-7
yi-2
yi-1
UNSW SYDNEY

19


Math board
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
$e_{ij} = a(s_{i-1}, h_j)$
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
$s_i = f(s_{i-1}, y_{i-1}, c_i)$
$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$
yi-7
yi-6
yi-1
yi
g
ci
αi-8,j
αi-7,j
αi-1,j
si-8
si-7
si-1
si
hj
yi-8
yi-7
yi-2
yi-1
UNSW SYDNEY

20


Math board
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
$e_{ij} = a(s_{i-1}, h_j)$
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
$s_i = f(s_{i-1}, y_{i-1}, c_i)$
$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$
y_{i-7} y_{i-6} y_{i-1} y_i
g
c_i
$\alpha_{i-8,j}$ $\alpha_{i-7,j}$ $\alpha_{i-1,j}$
s_{i-8} s_{i-7} s_{i-1} s_i
h_j
y_{i-8} y_{i-7} y_{i-2} y_{i-1}
The next word is generated as a combination of current state and context over all previous positions!
Context vector: A vector as a combination of all attention vectors.
Attention vectors: Vectors that capture association between current position and past positions.
State vectors: A vector that stores contextual information of past positions in the sentence.
Hidden state: A vector of the current position used to compare with past positions.
UNSW SYDNEY

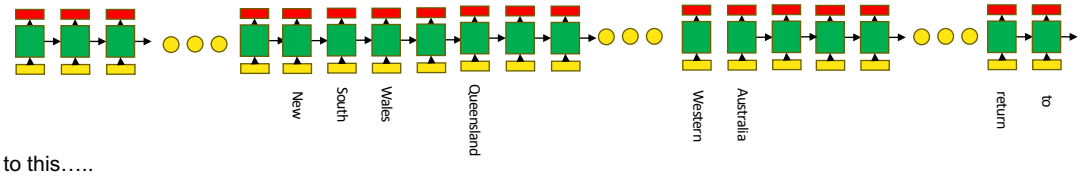
21


When you have to wait a minute
Math board
$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$
$e_{ij} = a(s_{i-1}, h_j)$
$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
$s_i = f(s_{i-1}, y_{i-1}, c_i)$
Wait a minute
made with mematic
t word is generated as a
ation of current state and
over all previous positions!
vector: A vector as a
ation of all attention vectors.
n vectors: Vectors that
association between current
and past positions.
ectors: A vector that stores
ual information of past
ns in the sentence.
state: A vector of the
position used to compare
st positions.
UNSW SYDNEY

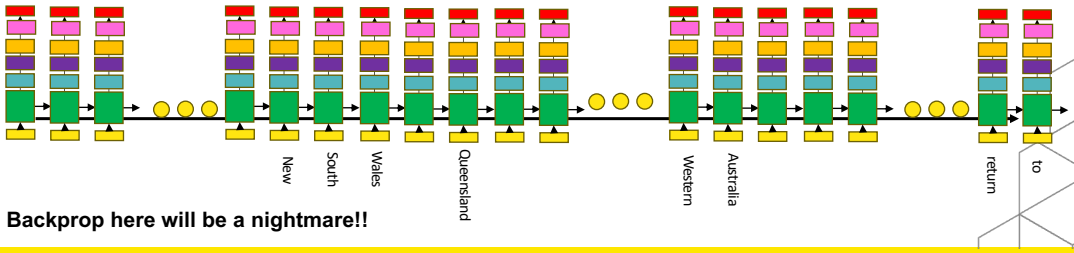
22

Where did this land us?

We went from this…



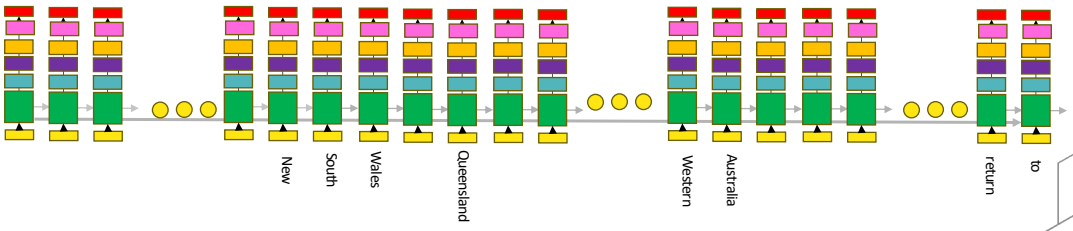
to this…..



Backprop here will be a nightmare!!

"Attention is all you need"

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).



"eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output"

A popular attention visualization

https://colab.research.google.com/drive/1S-Cnm5KpH5rjJXfDZH5flR0T58yT2dOF#scrollTo=rZgdi4xCK2X4

Source: https://github.com/jessevig/bertviz

Demo time!

25

UNSW SYDNEY | Australia's Global University

Part 2

Encoder & Decoder

26

Transformer

Transformer is a sequence-to-sequence (seq2seq) model that uses the attention mechanism

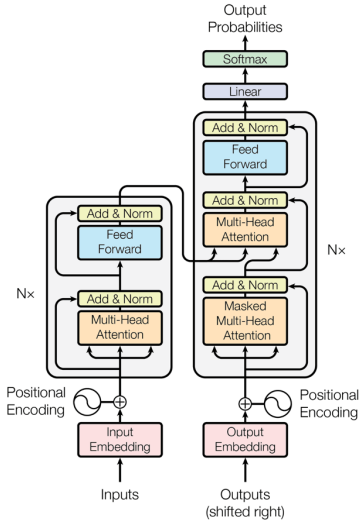
Transformer is a transducer that transforms a sentence into another

Originally evaluated on machine translation and other seq2seq tasks

Spawned several other kinds of NLP models (next module: encoder-only and decoder-only models)

Examples: BERT, PaLM, GPT, XLNet, OPT, BLOOM, etc.

… and non-NLP models



Components of a Transformer

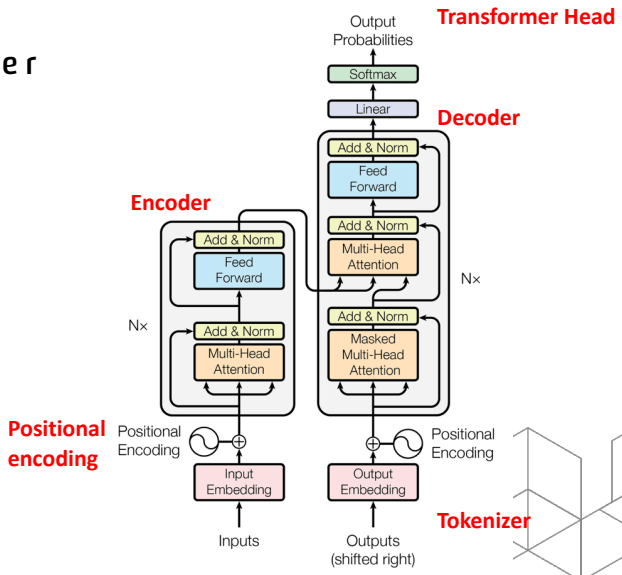
Encoder: A stack of encoders receive the input sentence and encode them into a hidden representation

Decoder: A stack of decoders receive the hidden representation from the encoder AND the text generated so far and produce the next token in the sequence.

Tokenizer: Represents a sentence as a set of tokens

Positional Encoding: Injects sequence information into the Transformer: a key reason why it can get rid of recurrence

Transformer Head: Attaches the prediction formulation used to optimize the model



https://www.researchgate.net/figure/The-total-number-of-publications-in-the-field-of-NLP-The-statistics-are-from-the-Web-of_fig1_372092521

Pseudocode time

Algorithm 8: $P \leftarrow \texttt{EDTransformer}(z, x|\theta)$

```
/* Encoder-decoder transformer forward pass                    */
```

Input: $z, x \in V^*$, two sequences of token IDs.

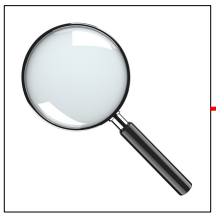
Output: $P \in (0,1)^{N_V \times \text{length}(x)}$, where the t-th column of P represents $\hat{P}_\theta(x[t+1]|x[1:t], z)$.

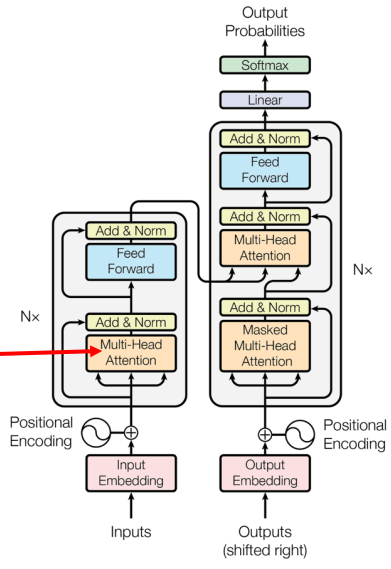
Hyperparameters: $\ell_{\max}, L_{\text{enc}}, L_{\text{dec}}, H, d_e, d_{\text{mlp}} \in \mathbb{N}$

z: Input; x: Output

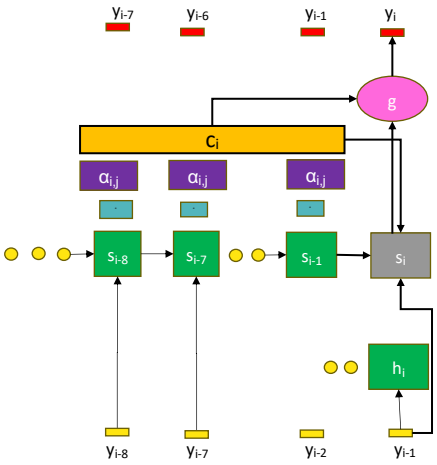
Next word in the output sequence; conditional on input sequence and the output sequence so far.

Let's go deep into the model





Attention is modelled as key-query-value pairs



The next word is generated as a combination of current state and context over all previous positions!

Context vector: A vector as a combination of all attention vectors. ↔ **Scaled dot-product attention**

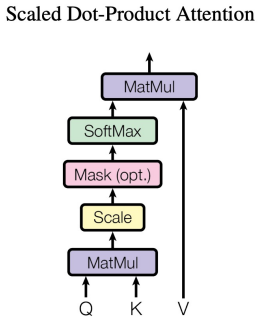
Attention vectors: Vectors that capture association between current position and past positions. ↔ **Key, Value:** Past state and corresponding vectors

State vectors: A vector that stores contextual information of past positions in the sentence.

Hidden state: A vector of the current position used to compare with past positions. ↔ **Query:** A vector of the current token

Scaled-dot product attention

Query, keys, values, and output are all vectors



Scaled dot-product attention

Dot-product

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

Scaled

.. what? THREE vectors?

…. and the 'supermarket' example!

Key	Value
Key 1	Value 1
Key 2	Value 2
..	…

A simple attention formulation

Many variants of attention

- Original formulation: $a(\mathbf{q}, \mathbf{k}) = w_2^T \tanh(W_1[\mathbf{q}; \mathbf{k}])$
- Bilinear product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T W \mathbf{k}$ Luong et al., 2015
- Dot product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k}$ Luong et al., 2015
- Scaled dot product: $a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\sqrt{|\mathbf{k}|}}$ Vaswani et al., 2017

Pen-and-paper time!

Advanced Reading (Optional for the course)

Nadaraya-Watson Estimator (1964)
https://d2l.ai/chapter_attention-mechanisms-and-transformers/attention-pooling.html

Credit: Mohit Iyyer, UMass

Pseudocode: Attention

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.

Parameters: $\boldsymbol{W_q}, \boldsymbol{W_k} \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b_q}, \boldsymbol{b_k} \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.

Parameters: $\boldsymbol{W_v} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b_v} \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

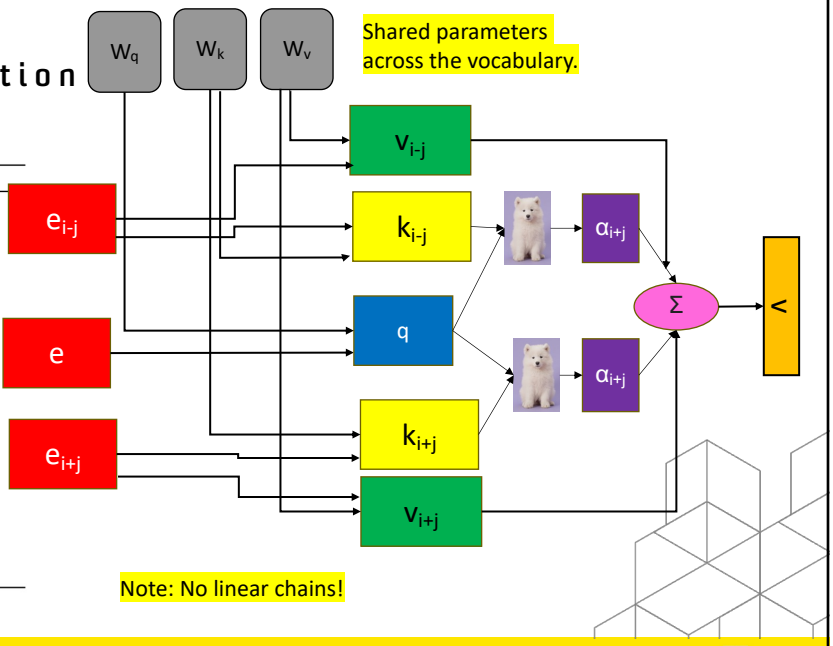
1 $\boldsymbol{q} \leftarrow \boldsymbol{W_q e} + \boldsymbol{b_q}$

2 $\forall t: \boldsymbol{k}_t \leftarrow \boldsymbol{W_k e}_t + \boldsymbol{b_k}$

3 $\forall t: \boldsymbol{v}_t \leftarrow \boldsymbol{W_v e}_t + \boldsymbol{b_v}$

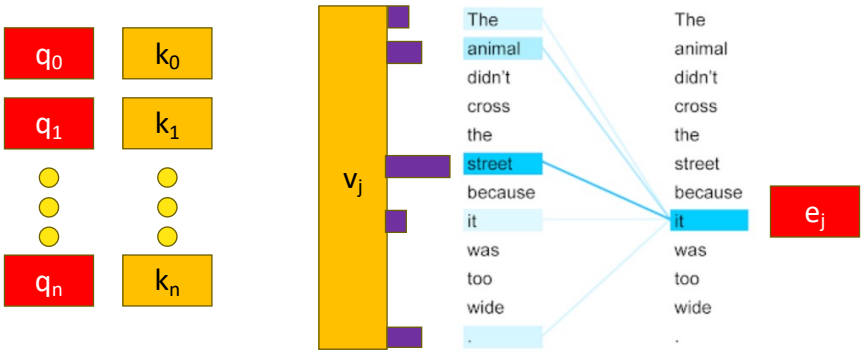
4 $\forall t: \alpha_t = \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_t / \sqrt{d_{\text{attn}}})}{\sum_u \exp(\boldsymbol{q}^\top \boldsymbol{k}_u / \sqrt{d_{\text{attn}}})}$

5 **return** $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$



Self-attention

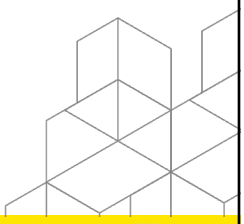
Self-attention: Attention for a token with respect to all other tokens in the sentence.



Semantics is multi-dimensional

I love nature: green trees, pink flowers, red apples

Remember syntagmatic and paradigmatic similarity?

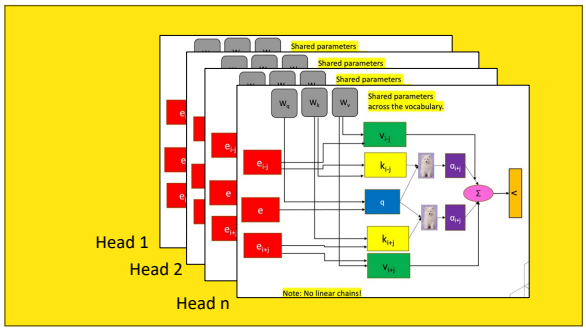


37

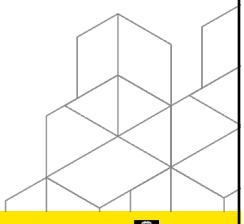
Multi-head attention

Can we learn multiple projections to capture different aspects of semantics?
They are independent. Therefore, can be parallelized.

Multi-head attention



The original Transformer paper uses 8 heads.



38

We need a way to combine these outputs....

Algorithm 6: $\hat{e} \leftarrow$ `layer_norm`$(e|\gamma,\beta)$

`/* Normalizes layer activations e. */`

Input: $e \in \mathbb{R}^{d_e}$, neural network activations.

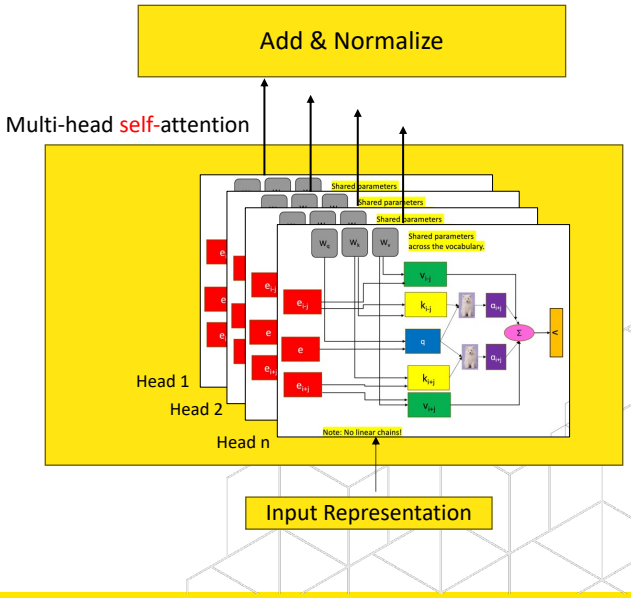
Output: $\hat{e} \in \mathbb{R}^{d_e}$, normalized activations.

Parameters: $\gamma, \beta \in \mathbb{R}^{d_e}$, element-wise scale and offset.

1 $m \leftarrow \sum_{i=1}^{d_e} e[i]/d_e$

2 $v \leftarrow \sum_{i=1}^{d_e} (e[i]-m)^2/d_e$

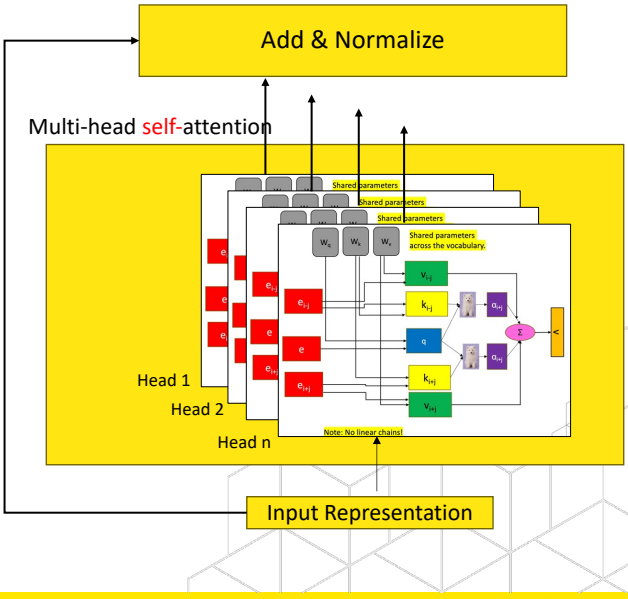
3 **return** $\hat{e} = \frac{e-m}{\sqrt{v}} \odot \gamma + \beta$, where $\odot$ denotes element-wise multiplication.



39

We need a way to combine these outputs....

...and some residual connections.



40

Encoders in Transformer

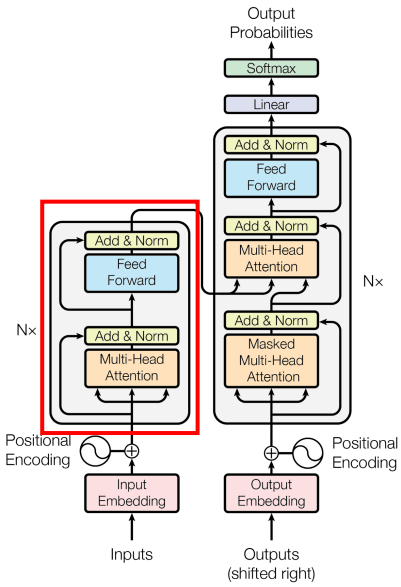
Each encoder:

(a) Multi-head self-attention

(b) Fully connected feed-forward network applied to every position separately

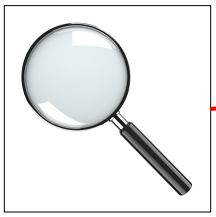
(c) Residual connection + Add-norm after (a) and (b)

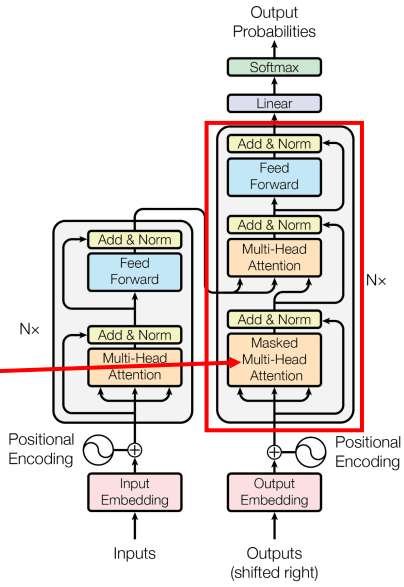
Transformer uses N = 6



41

Next: Decoders in Transformer



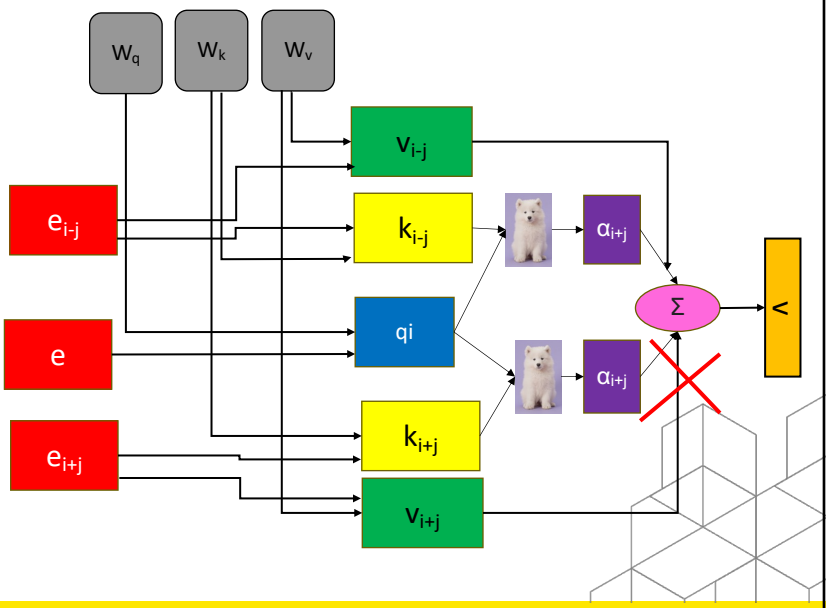


42

Masked attention

Can the decoder use subsequent tokens when generating a sequence?

Masked attention 'masks' the attention computation for subsequent tokens in a sequence.



Pseudocode: Single-head self-/cross-attention

Algorithm 3: Basic single-query attention.

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token
Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.
Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.
Parameters: $\boldsymbol{W_q}, \boldsymbol{W_k} \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, $\boldsymbol{b_q}, \boldsymbol{b_k} \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections.
Parameters: $\boldsymbol{W_v} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $\boldsymbol{b_v} \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection.

1 $\boldsymbol{q} \leftarrow \boldsymbol{W_q e} + \boldsymbol{b_q}$
2 $\forall t: \ \boldsymbol{k}_t \leftarrow \boldsymbol{W_k e}_t + \boldsymbol{b_k}$
3 $\forall t: \ \boldsymbol{v}_t \leftarrow \boldsymbol{W_v e}_t + \boldsymbol{b_v}$
4 $\forall t: \ \alpha_t = \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_t / \sqrt{d_{\text{attn}}})}{\sum_u \exp(\boldsymbol{q}^\top \boldsymbol{k}_u / \sqrt{d_{\text{attn}}})}$
5 **return** $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v}_t$

Algorithm 4: $\tilde{\boldsymbol{V}} \leftarrow \texttt{Attention}(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W_{qkv}}, \text{Mask})$

/* Computes a single (masked) self- or cross- attention head. */
Input: $\boldsymbol{X} \in \mathbb{R}^{d_x \times \ell_x}$, $\boldsymbol{Z} \in \mathbb{R}^{d_z \times \ell_z}$, vector representations of primary and context sequence.
Output: $\tilde{\boldsymbol{V}} \in \mathbb{R}^{d_{\text{out}} \times \ell_x}$, updated representations of tokens in $\boldsymbol{X}$, folding in information from tokens in $\boldsymbol{Z}$.
Parameters: $\boldsymbol{W_{qkv}}$ consisting of:
$\boldsymbol{W_q} \in \mathbb{R}^{d_{\text{attn}} \times d_x}$, $\boldsymbol{b_q} \in \mathbb{R}^{d_{\text{attn}}}$
$\boldsymbol{W_k} \in \mathbb{R}^{d_{\text{attn}} \times d_z}$, $\boldsymbol{b_k} \in \mathbb{R}^{d_{\text{attn}}}$
$\boldsymbol{W_v} \in \mathbb{R}^{d_{\text{out}} \times d_z}$, $\boldsymbol{b_v} \in \mathbb{R}^{d_{\text{out}}}$.
Hyperparameters: Mask$\in \{0,1\}^{\ell_z \times \ell_x}$, $\uparrow(3)$

1 $\boldsymbol{Q} \leftarrow \boldsymbol{W_q X} + \boldsymbol{b_q} \boldsymbol{1}^\top$ [[Query $\in \mathbb{R}^{d_{\text{attn}} \times \ell_x}$]]
2 $\boldsymbol{K} \leftarrow \boldsymbol{W_k Z} + \boldsymbol{b_k} \boldsymbol{1}^\top$ [[Key $\in \mathbb{R}^{d_{\text{attn}} \times \ell_z}$]]
3 $\boldsymbol{V} \leftarrow \boldsymbol{W_v Z} + \boldsymbol{b_v} \boldsymbol{1}^\top$ [[Value $\in \mathbb{R}^{d_{\text{out}} \times \ell_z}$]]
4 $\boldsymbol{S} \leftarrow \boldsymbol{K}^\top \boldsymbol{Q}$ [[Score $\in \mathbb{R}^{\ell_z \times \ell_x}$]]
5 $\forall t_z, t_x$, if $\neg$Mask$[t_z, t_x]$ then $S[t_z, t_x] \leftarrow -\infty$
6 **return** $\tilde{\boldsymbol{V}} = \boldsymbol{V} \cdot \text{softmax}\left(\boldsymbol{S} / \sqrt{d_{\text{attn}}}\right)$

Get the query, key, value combination

Compute the similarity score

For subsequent positions, mask the value.

Scaled dot-product attention

… and multi-head attention

Algorithm 5: $\tilde{V} \leftarrow \texttt{MHAttention}(X, Z|\mathcal{W}, \text{Mask})$

```
/* Computes Multi-Head (masked) self-
   or cross- attention layer.        */
```

Input: $X \in \mathbb{R}^{d_x \times \ell_x}$, $Z \in \mathbb{R}^{d_z \times \ell_z}$, vector representations of primary and context sequence.

Output: $\tilde{V} \in \mathbb{R}^{d_{\text{out}} \times \ell_x}$, updated representations of tokens in X, folding in information from tokens in Z.

Hyperparameters: H, number of attention heads

Hyperparameters: Mask$\in\{0,1\}^{\ell_z \times \ell_x}$, ↑(3)

Parameters: $\mathcal{W}$ consisting of

For $h \in [H]$, $\mathcal{W}^h_{qkv}$ consisting of:

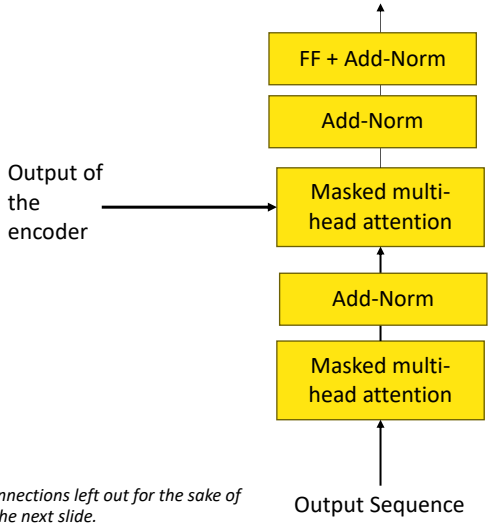
- $W^h_q \in \mathbb{R}^{d_{\text{attn}} \times d_x}$, $b^h_q \in \mathbb{R}^{d_{\text{attn}}}$,
- $W^h_k \in \mathbb{R}^{d_{\text{attn}} \times d_z}$, $b^h_k \in \mathbb{R}^{d_{\text{attn}}}$,
- $W^h_v \in \mathbb{R}^{d_{\text{mid}} \times d_z}$, $b^h_v \in \mathbb{R}^{d_{\text{mid}}}$.

$W_o \in \mathbb{R}^{d_{\text{out}} \times H d_{\text{mid}}}$, $b_o \in \mathbb{R}^{d_{\text{out}}}$.

For $h \in [H]$:
  $Y^h \leftarrow \texttt{Attention}(X, Z|\mathcal{W}^h_{qkv}, \text{Mask})$
$Y \leftarrow [Y^1; Y^2; \ldots; Y^H]$
return $\tilde{V} = W_o Y + b_o \mathbf{1}^\intercal$

h iterations that run in parallel

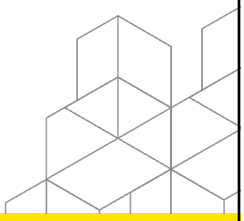
Let's construct the decoder



Attention over the output sequence relative to the output of the encoder.

Attention over the output sequence generated so far.

.. residual connections left out for the sake of brevity. See the next slide.



Decoders in Transformer

Each decoder:

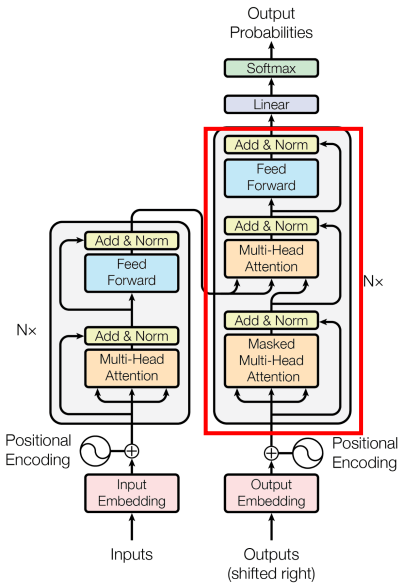
(a) Masked Multi-head attention

(b) Fully connected feed-forward network applied to every position separately

(c) Residual connection + Add-norm after (a) and (b)

(d) Multi-head attention (over the output of the encoder)

Transformer uses N = 6



Transformer Training

Algorithm 11: $\hat{\boldsymbol{\theta}} \leftarrow \texttt{EDTraining}(\boldsymbol{z}_{1:N_{\text{data}}}, \boldsymbol{x}_{1:N_{\text{data}}}, \boldsymbol{\theta})$

```
/* Training a seq2seq model */
```

Input: $\{(\boldsymbol{z}_n, \boldsymbol{x}_n)\}_{n=1}^{N_{\text{data}}}$, a dataset of sequence pairs.
Input: $\boldsymbol{\theta}$, initial transformer parameters.
Output: $\hat{\boldsymbol{\theta}}$, the trained parameters.
Hyperparameters: $N_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$

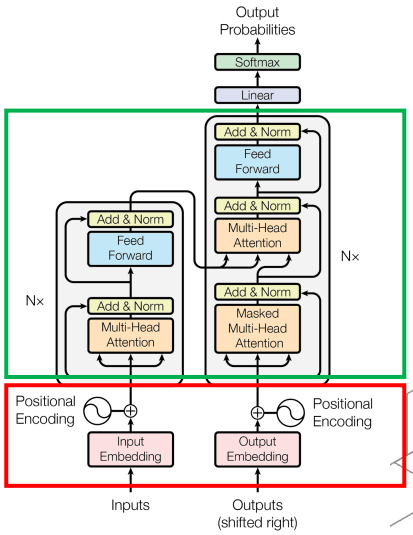
for $i = 1, 2, \ldots, N_{\text{epochs}}$ **do**
  for $n = 1, 2, \ldots N_{\text{data}}$ **do**
    $\ell \leftarrow \text{length}(\boldsymbol{x}_n)$
    $\boldsymbol{P}(\boldsymbol{\theta}) \leftarrow \texttt{EDTransformer}(\boldsymbol{z}_n, \boldsymbol{x}_n | \boldsymbol{\theta})$
    $\text{loss}(\boldsymbol{\theta}) = -\sum_{t=1}^{\ell-1} \log P(\boldsymbol{\theta})[x_n[t+1], t]$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla \text{loss}(\boldsymbol{\theta})$
  end
end
return $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$

Part 3

Tokenization and Positional Encoding

49

Let's now look at the input representations



50

Tokenization

Tokenization: The process of splitting a sequence into tokens

Representations will be learned for tokens

Types of tokenization	Input	Output
Word-level tokenization	"Are you feeling sleepy?"	Are, you, feeling, sleepy
Character-level tokenization		A, r, e, , y,o……
Sub-word tokenization		A, #r, #e, you, feel, #ing, sleep#, #y

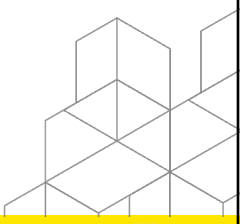
Remember stemming and lemmatization?

Sub-word tokenization

Example: Byte-pair encoding

Originally developed as a text compression algorithm

Also, WordPiece Tokenizer: https://www.youtube.com/watch?v=qpv6ms_t_1A



UNSW SYDNEY

51

Byte-pair encoding algorithm

Intuition:

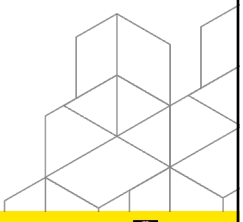
Retain the most common words as tokens

Break the less common words into sub-word tokens

Why?

Each token will be represented by an ID. Common words will be represented wholly. See how this relates to text compression?

Algorithm: Corpus—based tokenizer

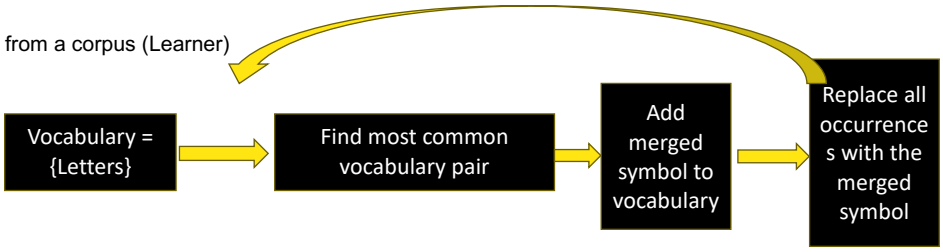


UNSW SYDNEY

52

Corpus-based tokenization

Step 1: Learn tokenization from a corpus (Learner)



Store new symbols in the dictionary

Save merge rules

Step 2: Use the learned tokenizer on test sentences (Tokenizer)

53

54

Tokenizer

Demo time!

Token embedding

Once input has been split into tokens, token embeddings are learned during training.
Therefore, during inference:

Algorithm 1: Token embedding.

Input: $v \in V \cong [N_V]$, a token ID.
Output: $\boldsymbol{e} \in \mathbb{R}^{d_e}$, the vector representation of the token.
Parameters: $\boldsymbol{W_e} \in \mathbb{R}^{d_e \times N_V}$, the token embedding matrix.

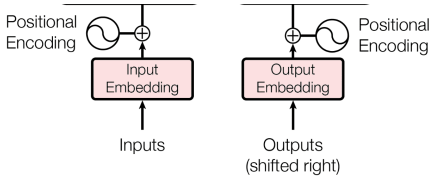
1 **return** $\boldsymbol{e} = \boldsymbol{W_e}[:, v]$

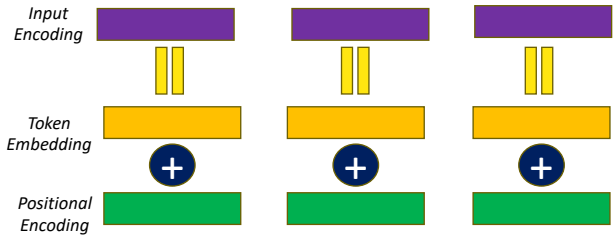
What next…

Tokens have an embedding

….but remember that there's no recurrence.

So, Transformer 'injects' a positional encoding.



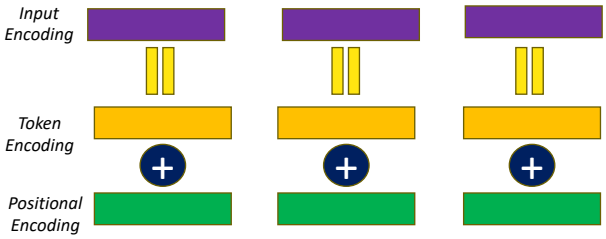


57

Positional encoding

Vectors corresponding to each position in a sequence.

Assume token embeddings are of length 50.



If max_length = 25, we would like to have 25 position vectors of length 50, such that…

- Each position vector is unique
- Should generalize to longer lengths

Original Transformer paper uses fixed positional encoding. Learnable encodings used in newer adaptations.

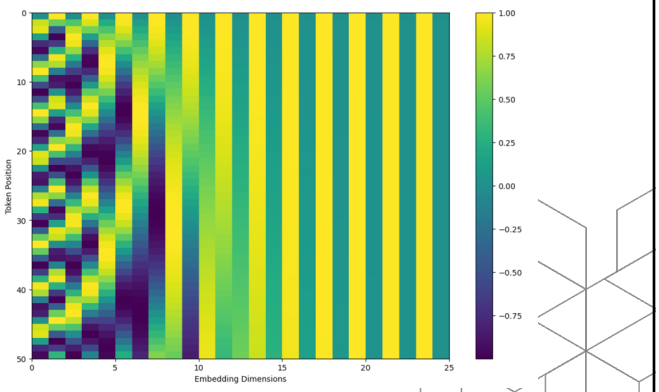
58

Fixed positional encoding

"we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, P Epos+k can be represented as a linear function of PEpos."

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Demo time!



59

UNSW SYDNEY | Australia's Global University

Part 3

Transformer: Looking Back

60

Quiz: Match textual descriptors (A)-(D) with one or more of (1)-(8) in the diagram

Example: Textual sequence that is input to the Transformer: 8

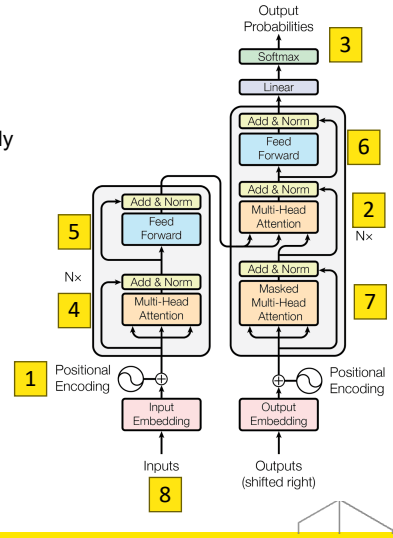
(A) Implemented as fixed embeddings

(B) Attention module that looks at preceding tokens in the sequence only

(C) Language model head that predicts the output sequence

(D) Uses key, query and values

Write all that apply.



61

Quiz: Match textual descriptors (A)-(D) with one or more of (1)-(8) in the diagram

Example: Textual sequence that is input to the Transformer: 8

(A) Implemented as fixed embeddings : [1]

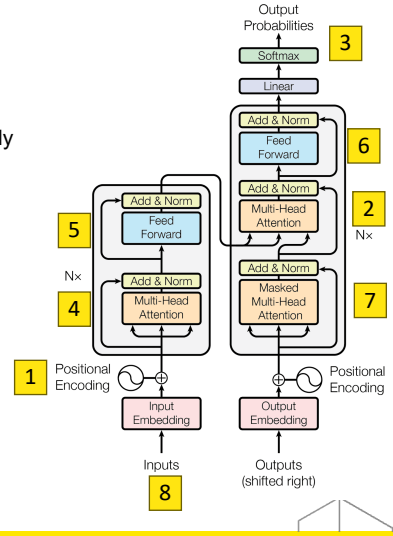
(B) Attention module that looks at preceding tokens in the sequence only

[7]

(C) Language model head that predicts the output sequence: 3

(D) Uses key, query and values: 2, 4, 7

Select all that apply.



62

Let's look at the pseudocode of the Transformer

63

Specification: Reminder & the full thing

Algorithm 8: $P \leftarrow \texttt{EDTransformer}(z, x|\theta)$

```
/* Encoder-decoder transformer forward pass                    */
```

Input: $z, x \in V^*$, two sequences of token IDs.
Output: $P \in (0,1)^{N_V \times \text{length}(x)}$, where the t-th column of P represents $\hat{P}_\theta(x[t+1]|x[1:t], z)$.
Hyperparameters: $\ell_{\max}, L_{\text{enc}}, L_{\text{dec}}, H, d_e, d_{\text{mlp}} \in \mathbb{N}$
Parameters: θ includes all of the following parameters:
$W_e \in \mathbb{R}^{d_e \times N_V}$, $W_p \in \mathbb{R}^{d_e \times \ell_{\max}}$, the token and positional embedding matrices.
For $l \in [L_{\text{enc}}]$:
| W_l^{enc}, multi-head encoder attention parameters for layer l, see (4),
| $\gamma_l^1, \beta_l^1, \gamma_l^2, \beta_l^2 \in \mathbb{R}^{d_e}$, two sets of layer-norm parameters,
| $W_{\text{mlp1}}^l \in \mathbb{R}^{d_{\text{mlp}} \times d_e}$, $b_{\text{mlp1}}^l \in \mathbb{R}^{d_{\text{mlp}}}$, $W_{\text{mlp2}}^l \in \mathbb{R}^{d_e \times d_{\text{mlp}}}$, $b_{\text{mlp2}}^l \in \mathbb{R}^{d_e}$, MLP parameters.
For $l \in [L_{\text{dec}}]$:
| W_l^{dec}, multi-head decoder attention parameters for layer l, see (4),
| $W_l^{e/d}$, multi-head cross-attention parameters for layer l, see (4),
| $\gamma_l^3, \beta_l^3, \gamma_l^4, \beta_l^4, \gamma_l^5, \beta_l^5 \in \mathbb{R}^{d_e}$, three sets of layer-norm parameters,
| $W_{\text{mlp3}}^l \in \mathbb{R}^{d_{\text{mlp}} \times d_e}$, $b_{\text{mlp3}}^l \in \mathbb{R}^{d_{\text{mlp}}}$, $W_{\text{mlp4}}^l \in \mathbb{R}^{d_e \times d_{\text{mlp}}}$, $b_{\text{mlp4}}^l \in \mathbb{R}^{d_e}$, MLP parameters.
$W_u \in \mathbb{R}^{N_V \times d_e}$, the unembedding matrix.

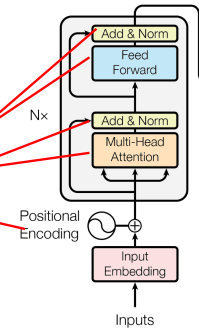
z: Input; x: Output

Next word in the output sequence; conditional on Next word in the sequence output sequence; ə output conditional on uence so far. input sequence and the output sequence so far.

64

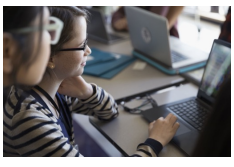
Encoder

/* Encode the context sequence:
$\ell_z \leftarrow \text{length}(\boldsymbol{z})$
for $t \in [\ell_z]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W}_e[:, z[t]] + \boldsymbol{W}_p[:, t]$
$\boldsymbol{Z} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_{\ell_z}]$
for $l = 1, 2, \ldots, L_{\text{enc}}$ **do**
  $\boldsymbol{Z} \leftarrow \boldsymbol{Z} + \texttt{MHAttention}(\boldsymbol{Z} | \boldsymbol{W}_l^{\text{enc}}, \text{Mask} \equiv 1)$
  for $t \in [\ell_z]$: $\boldsymbol{Z}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{Z}[:, t] | \gamma_l^1, \beta_l^1)$
  $\boldsymbol{Z} \leftarrow \boldsymbol{Z} + \boldsymbol{W}^l_{\text{mlp2}} \text{ReLU}(\boldsymbol{W}^l_{\text{mlp1}} \boldsymbol{Z} + \boldsymbol{b}^l_{\text{mlp1}} \boldsymbol{1}^\top) + \boldsymbol{b}^l_{\text{mlp2}} \boldsymbol{1}^\top$
  for $t \in [\ell_z]$: $\boldsymbol{Z}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{Z}[:, t] | \gamma_l^2, \beta_l^2)$
end

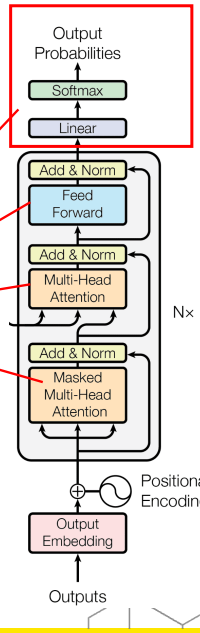


Decoder

/* Decode the primary sequence, conditioning on the context:
$\ell_x \leftarrow \text{length}(\boldsymbol{x})$
for $t \in [\ell_x]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W}_e[:, x[t]] + \boldsymbol{W}_p[:, t]$
$\boldsymbol{X} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_{\ell_x}]$
for $i = 1, 2, \ldots, L_{\text{dec}}$ **do**
  $\boldsymbol{X} \leftarrow \boldsymbol{X} + \texttt{MHAttention}(\boldsymbol{X} | \boldsymbol{W}_l^{\text{dec}}, \text{Mask}[t, t'] \equiv [[t \le t']])$
  for $t \in [\ell_x]$: $\boldsymbol{X}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{X}[:, t] | \gamma_l^3, \beta_l^3)$
  $\boldsymbol{X} \leftarrow \boldsymbol{X} + \texttt{MHAttention}(\boldsymbol{X}, \boldsymbol{Z} | \boldsymbol{W}_l^{\text{e/d}}, \text{Mask} \equiv 1)$
  for $t \in [\ell_x]$: $\boldsymbol{X}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{X}[:, t] | \gamma_l^4, \beta_l^4)$
  $\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{W}^l_{\text{mlp4}} \text{ReLU}(\boldsymbol{W}^l_{\text{mlp3}} \boldsymbol{X} + \boldsymbol{b}^l_{\text{mlp3}} \boldsymbol{1}^\top) + \boldsymbol{b}^l_{\text{mlp4}} \boldsymbol{1}^\top$
  for $t \in [\ell_x]$: $\boldsymbol{X}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{X}[:, t] | \gamma_l^5, \beta_l^5)$
end
/* Derive conditional probabilities and return:
return $\boldsymbol{P} = \text{softmax}(\boldsymbol{W}_u \boldsymbol{X})$



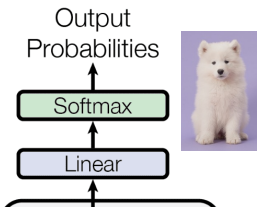
Demo time!



Final note: The model head

Model head
A linear layer that predicts probability over the vocabulary.
Loss gets calculated at this stage

"Language model head is the circuitry we do to do language model



You will see the model head in Week 4 when we discuss BERT and GPT.

Configuration Parameters

Vocabulary: 37000 tokens
8 NVIDIA P100 GPUs
d_model = 512
N = 6
Trained on the English-to-German, English-to-French WMT benchmark.

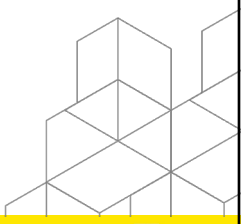
It is customary to include configuration parameters in research papers to ensure reproducibility.

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

BLEU: Higher the better. FLOPs: Lower the better



69

Let's look at another view of Transformer.

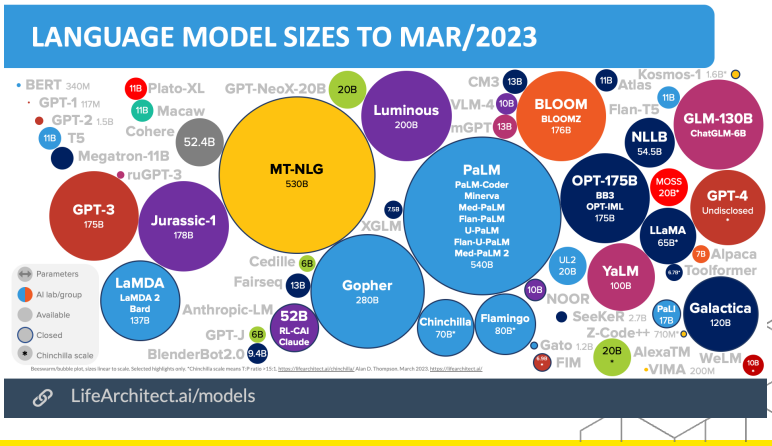
https://jalammar.github.io/illustrated-transformer/

70

The impact of Transformer on NLP

Transformer ushered a new revolution in NLP models

Advent of 'large' language models



https://www.fullydistributed.co/p/language-models-size-matters

The impact of Transformer on the world

LLMs are not going to destroy the human race

It's just a chatbot, dude.

NOAH SMITH
8 MAR 2023

137 220 Share

"This is the sort of thing you lifeforms enjoy, is it?" — Marvin the Paranoid Android

Five years ago, if you told me that in 2023 I would be writing a post arguing that chatbots are not likely to wipe out humanity, I would have said "...Yeah, sounds about right." Not because I knew anything about progress in large language models five years ago, but because on this crazy internet of ours, it's always something.

ARTIFICIAL INTELLIGENCE / TECH / US & WORLD

The EU AI Act passed – now comes the waiting

/ Delays in implementing the AI Act means nothing changes for now.

By Emilia David, a reporter who covers AI. Prior to joining The Verge, she covered the intersection between technology, finance, and the economy.

Dec 15, 2023, 8:47 AM GMT+11

Illustration by Alex Castro / The Verge

[1] Blog by Noah Smith
[2] https://www.theverge.com/2023/12/14/24001919/eu-ai-act-foundation-models-regulation-data

Many types of attentions….

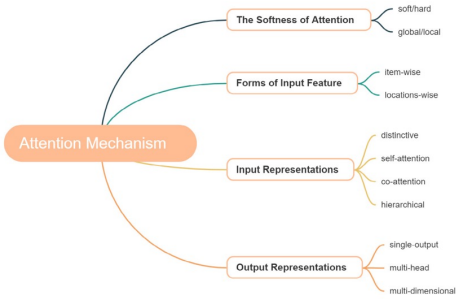


Fig. 1. Several typical approaches to attention mechanisms.

Table 1
Summary of score function f. Here, $\boldsymbol{k}$ is an element of $\boldsymbol{K}$, $\boldsymbol{v}, \boldsymbol{b}, \boldsymbol{W}, \boldsymbol{W_1}, \boldsymbol{W_2}$ are learnable parameters, d_k is the dimension of the input vector. The *act* is a nonlinear activation function, such as tanh and ReLU.

Name	Equation	Ref.
Additive	$f(q,k) = \boldsymbol{v}^\top act(W_1 k + W_2 q + b)$	[15]
Multiplicative (dot-product)	$f(q,k) = q^\top k$	[15]
Scaled multiplicative	$f(q,k) = \frac{q^\top k}{\sqrt{d_k}}$	[16]
General	$f(q,k) = q^\top W k$	[14]
Concat	$f(q,k) = \boldsymbol{v}^\top act(W[k;q] + b)$	[14]
Location-based	$f(q,k) = f(q)$	[14]
Similarity	$f(q,k) = \frac{q \cdot k}{\|q\| \cdot \|k\|}$	[60]

Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning." *Neurocomputing* 452 (2021): 48-62.

73

Many types of position encodings….

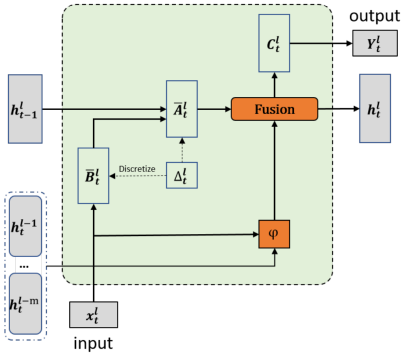
Data Structure	Model	Ref. Point	Inject. Met.	Learnable	Recurring	Unbound	#Param
Sequence	Transformer w/ emb. (Vaswani et al. 2017) BERT (Devlin et al. 2019) Reformer (Kitaev, Kaiser, and Levskaya 2020)	A	APE	✔	✖	✖	$t_{max} d$ $(d-d_t)^{t_{max}}_{t_1} + d_1 t_1$
	FLOATER (Liu et al. 2020)	A	APE	✔	✔	✔	0 or more
	Shortformer (Press, Smith, and Lewis 2021)	A	APE	✖	✔	✔	0
	Wang et al. (2020)	A	–	✔	✖	✔	2nd
	Shaw, Uszkoreit, and Vaswani (2018) (abs)	A	MAM	✔	✔	✖	$2l^2_{max} dl$
	Shaw, Uszkoreit, and Vaswani (2018) (rel) T5 (Raffel et al. 2020) Huang et al. (2020)	R	MAM	✔	✔	✖	$2(2l_{max}-1)dl$ $(2l_{max}-1)h$ $dlh(2l_{max}-1)$
	DeBERTa (He et al. 2021)	B	Both	✔	✔	✖	$3t_{max} d$
	Transformer XL (Dai et al. 2019) TENER (Yan et al. 2019) DA-Transformer (Wu, Wu, and Huang 2021)	R	MAM	✔	✔	✔	$2d + d^2 lh$ $2dlh$ $2h$
	TUPE (Ke, He, and Liu 2021)	B	MAM	✔	✖	✖	$2d^2 + t_{max}(d+2)$
	RNN-Transf. (Neishi and Yoshinaga 2019)	R	–	✔	✖	✔	$6d^2 + 3d$
	SPE (Liutkus et al. 2021)	R	MAM	✔	✔	✖	$3Kdh + ld$
	Transformer w/ sin. (Vaswani et al. 2017) Li et al. (2019) Takase and Okazaki (2019) Oka et al. (2020)	A	APE	✖	✖	✔	0
	Universal Transf. (Dehghani et al. 2019)	A	APE	✖	✔	✔	0
	DISAN (Shen et al. 2018) Rotary (Su et al. 2021)	R	MAM	✖	✔	✔	0
Tree	SPR-abs (Wang et al. 2019)	A	APE	✖	✖	✔	0
	SPR-rel (Wang et al. 2019)	R	MAM	✔	✖	✖	$2(2l_{max}+1)d$
	TPE (Shiv and Quirk 2019)	A	APE	✔	✖	✖	$\frac{d}{D_{max}}$
Graph	Struct. Transformer (Zhu et al. 2019) Graph Transformer (Cai and Lam 2020)	R	MAM	✔	✔	✔	$5d^2 + (d+1)d_e$ $7d^2 + 3d$
	Graformer (Schmitt et al. 2021)	R	MAM	✔	✔	✖	$2(D_{max}+1)h$
	Graph Transformer (Dwivedi and Bresson 2020) GRAPH-BERT (Zhang et al. 2020)	A B	APE	✖	✖	✔	0

Dufter, P., Schmitt, M., & Schütze, H. (2022). Position information in transformers: An overview. *Computational Linguistics*, *48*(3), 733-763.

74

Advanced Reading:

... and beyond Transformer



(a) DenseSSM in autoregressive mode.

He, Wei, Kai Han, Yehui Tang, Chengcheng Wang, Yujie Yang, Tianyu Guo, and Yunhe Wang. "Densemamba: State space models with dense hidden connection for efficient large language models." *arXiv preprint arXiv:2403.00818* (2024).

75

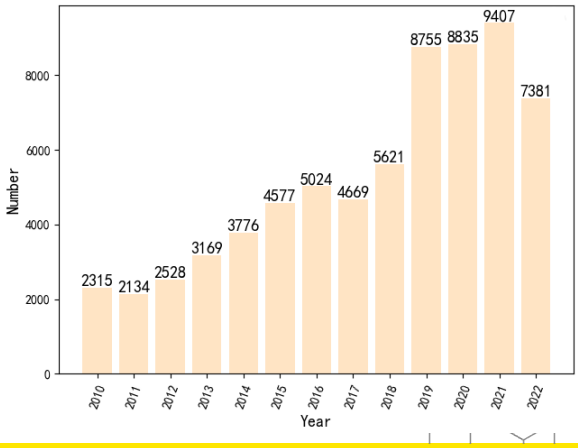
Surge in NLP research (NOT all from Big Tech)...
...and a 9-week course

This 'introductory' course is a distilled version of NLP.
Why do you want to study NLP?

or

(Reminder: Lecture 1: "Course Philosophy"; "Why are we even learning pre-deep learning NLP?")

Try: https://saifmohammad.com/WebPages/nlpscholar.html



https://www.researchgate.net/figure/The-total-number-of-publications-in-the-field-of-NLP-The-statistics-are-from-the-Web-of_fig1_372092521

76

Summary

Part	Key Idea	Demos
Why Attention?	Intuition behind attention	Pytorch tutorial
Attention in recurrent language modeling	Additive attention in recurrent LM and limitations	Attention Visualization
Encoders & decoders	Scaled dot-product attention -> Multi-head attention -> Self-attention, masked attention -> Encoders and decoders. Pseudocode + Animation.	Transformer tutorial
Tokenization and positional encoding	BPE algorithm, positional encoding, language model head	Transformer tutorial continued. BPE Tokenizer. Positional encoding Visualization
Transformer: Looking Back	Pseudocode; components; impact of Transformer.	

77

Transformer is the basis of the state-of-the-art models in NLP

New models continue to be released – one 'larger' than the previous.

Transformer spawned several derivative models…

'Large language models'

(Next week!)

78

Suggested Reading

All pseudocodes from: Phuong et al., Formal Algorithms for Transformers, https://arxiv.org/abs/2207.09238.

Annotated Transformer: https://jalammar.github.io/illustrated-transformer/

Byte-pair encoding: https://huggingface.co/learn/nlp-course/en/chapter6/5

Video: https://www.youtube.com/watch?v=zduSFxRajkE

Visualisation of Transformer (in Keras): https://colab.research.google.com/github/tensorflow/text/blob/master/docs/tutorials/transformer.ipynb

I hope the above links lead you into a rabbit-hole of interesting material. Do share it on the forum with the rest of your class.

So what kind of models did Transformer lead to?

➔ Large Language Models (Next week!)