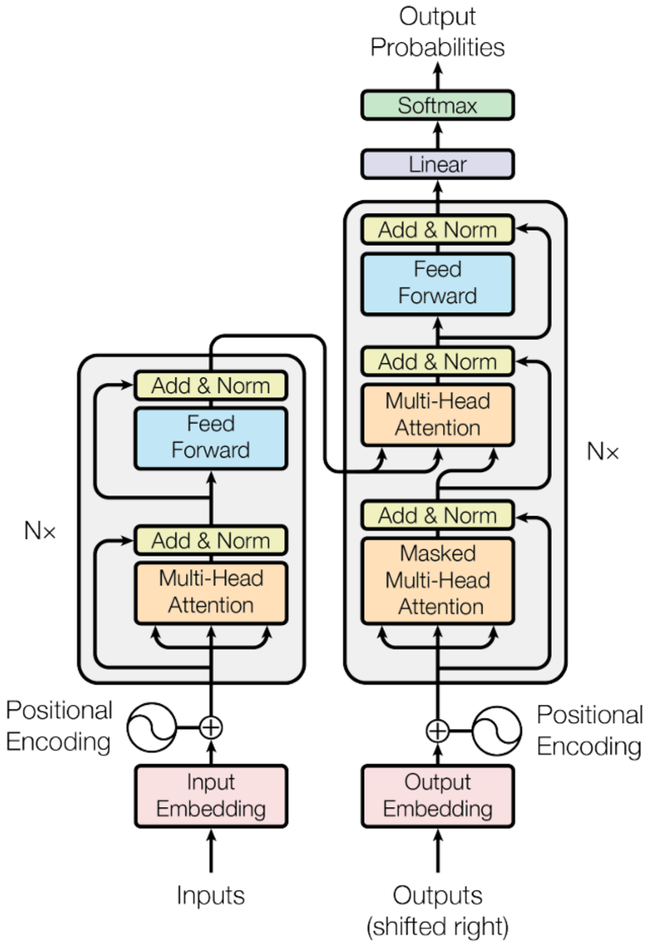
Machine Learning I
60629A

Attention and Transformers
— Week #10


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Transformers

- A deep learning model
  - Introduced in 2017 (Google researchers)
    - Quickly adopted for modelling sequential data (text and images)—architecture behind LLMs
  - Uses the *attention mechanism*

* Most of the slides/figures/narrative are from David Berger (you might see them again in ML #2).

Today’s plan

- Refresh our understanding of RNNs and bidirectional ones
- Introduce *attention*
- Transformer block
- Examples of transformers in practice (more on that next week)

- For concreteness, I will think of our data as a sequence of words
- And I use words and tokens synonymously
- In practice, tokens are sub-words (e.g., a few letters)
  - Tokenization is a topic beyond today's class

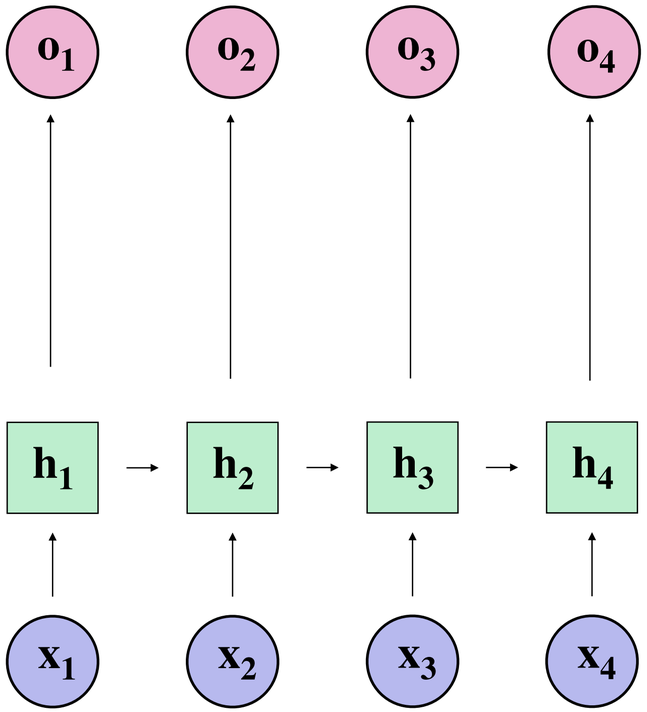
RNNs

For one time step

x_i Data

h_i Hidden layer (Left to right)

o_i Prediction



The hidden state at each timestep (h_t)
Must contain all useful information up to time t.

Bidirectional RNNs

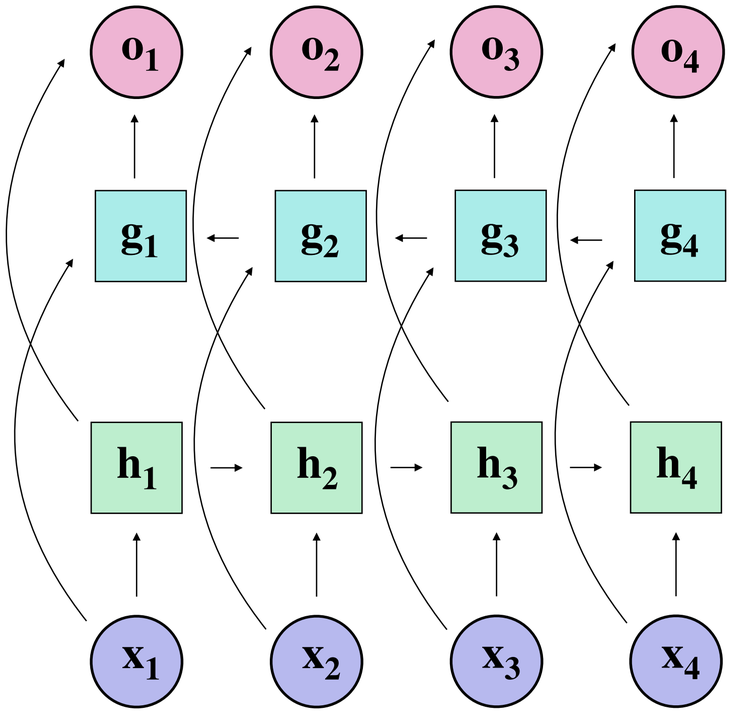
For one time step

x_i Data

h_i Hidden layer (Left to right)

g_i Hidden layer (Right to left)

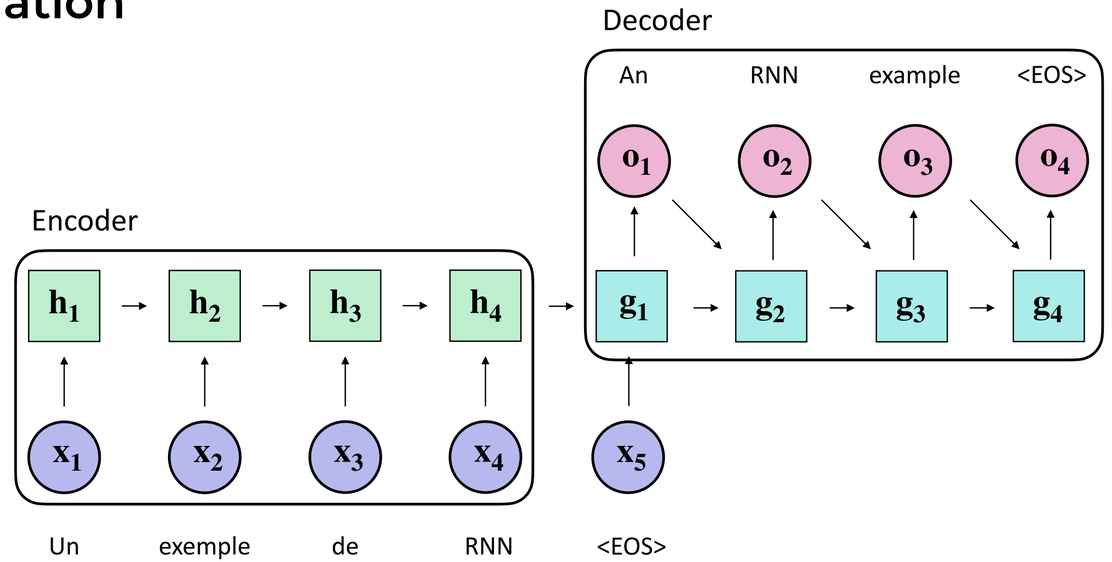
o_i Prediction



- The hidden state h_t must contain all useful information from to t
- The hidden state g_t must contain all useful information from the end (T) to t.
- Difficulties:
  - Exploding & vanishing gradients
    - Predictions will tend to use information from close neighbours

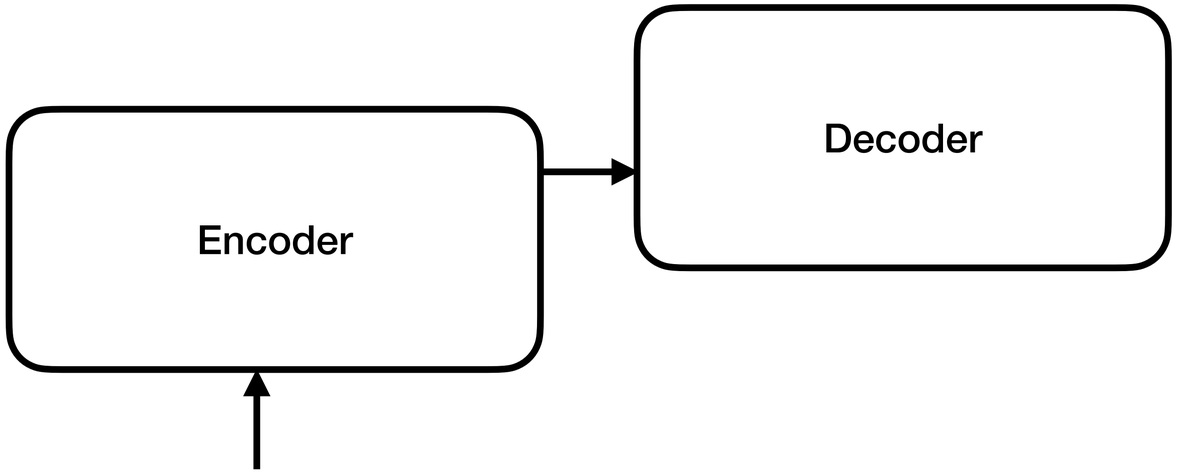
Encoder-Decoder
(Also known as Sequence to Sequence — Seq2Seq)

- What if the output is a different length than the input? E.g., translation



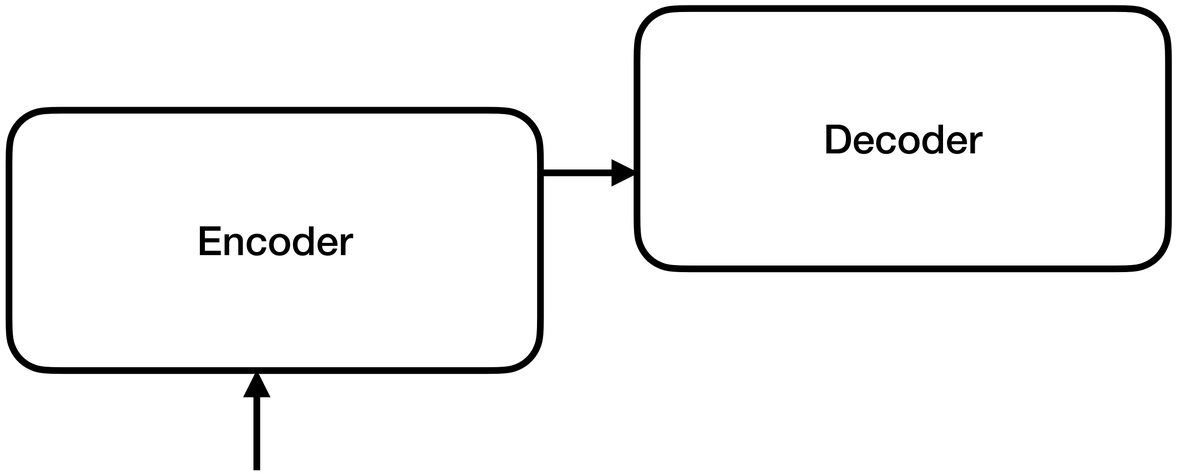
- h_4 must contain ("summarize") information from the input
  - It's a bottleneck. Its size (num. neurons) is an important hyperparameter

Problem



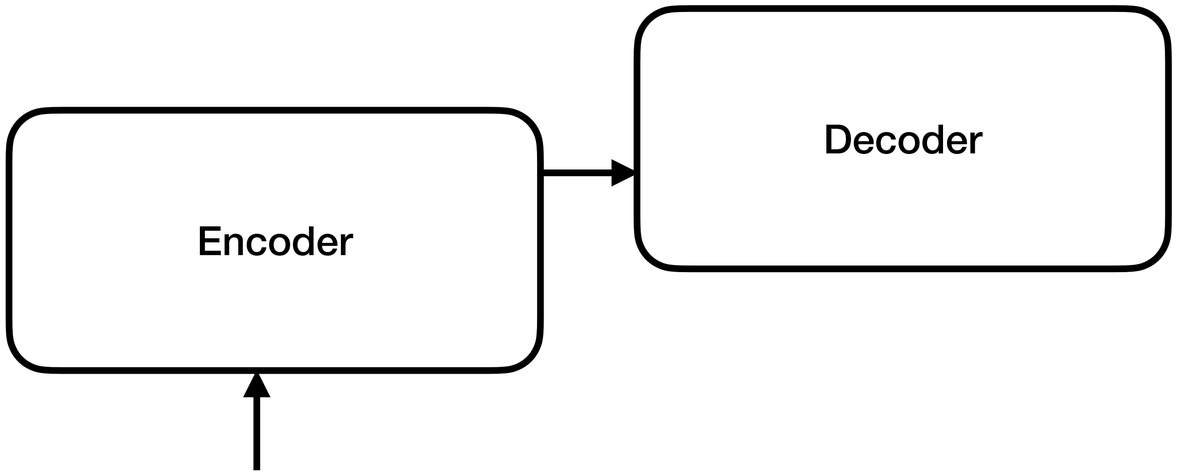
William Shakespeare (c. 23[a] April 1564 – 23 April 1616)[b] was an English playwright, poet and actor.
He is widely regarded as the greatest writer in the English language and the world's pre-eminent dramatist.[3][4][5]
He is often called England's national poet and the "Bard of Avon" (or simply "the Bard").
His extant works, including collaborations, consist of some 39 plays, 154 sonnets,
three long narrative poems and a few other verses, some of uncertain authorship.

Problem



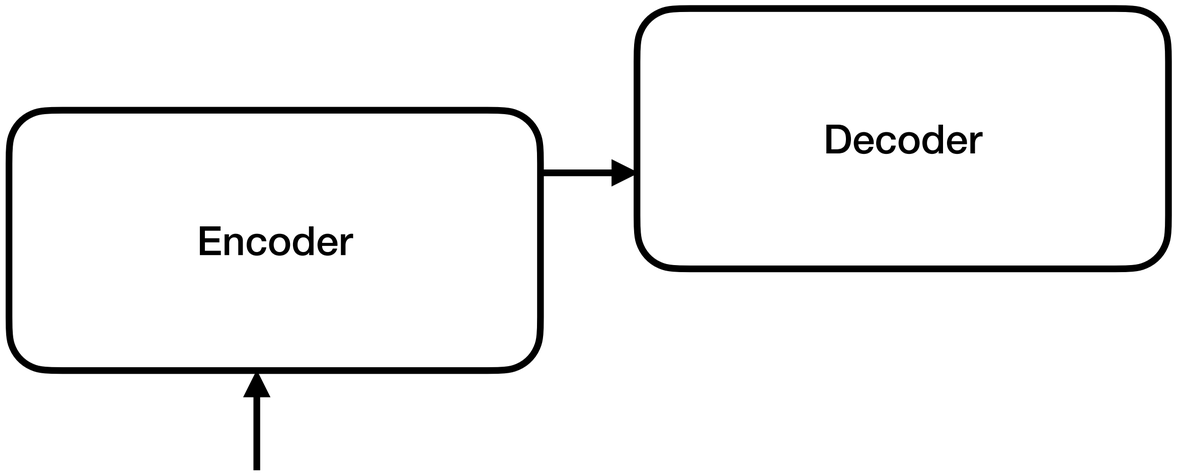
- **Challenging to encode vast amounts of information**

Problem



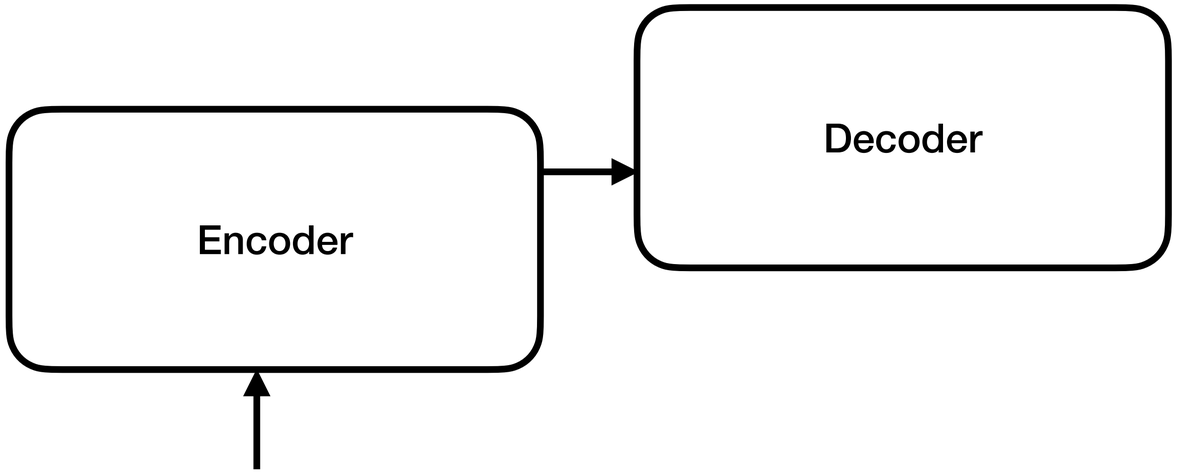
- **Challenging to encode vast amounts of information**
- **Could try to divide your input (e.g. sentence by sentence)**

Problem



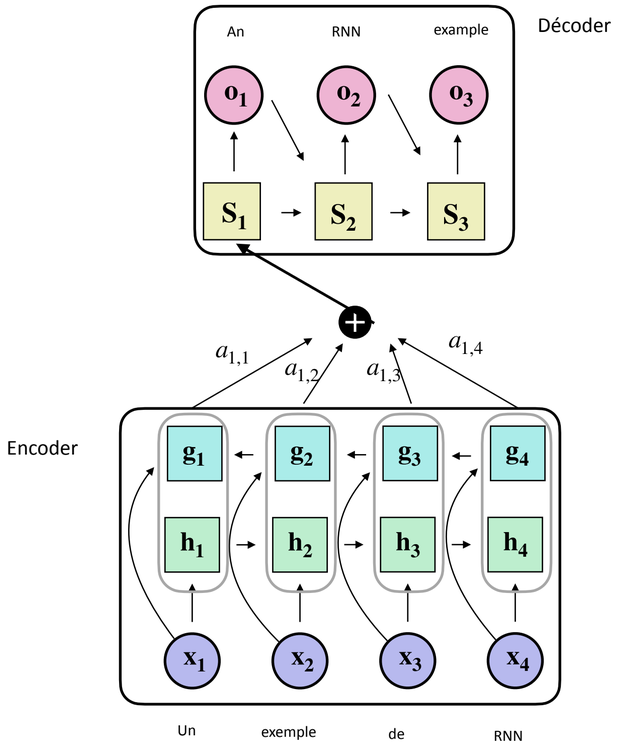
- Challenging to encode vast amounts of information
- Could try to divide your input (e.g. sentence by sentence)
  - Tasks (e.g. translation) can require *local* and *global* coherence

Problem



- Challenging to encode vast amounts of information
- Could try to divide your input (e.g. sentence by sentence)
  - Tasks (e.g. translation) can require *local* and *global* coherence
- Instead, decode word by word by focusing on the relevant different parts of the input

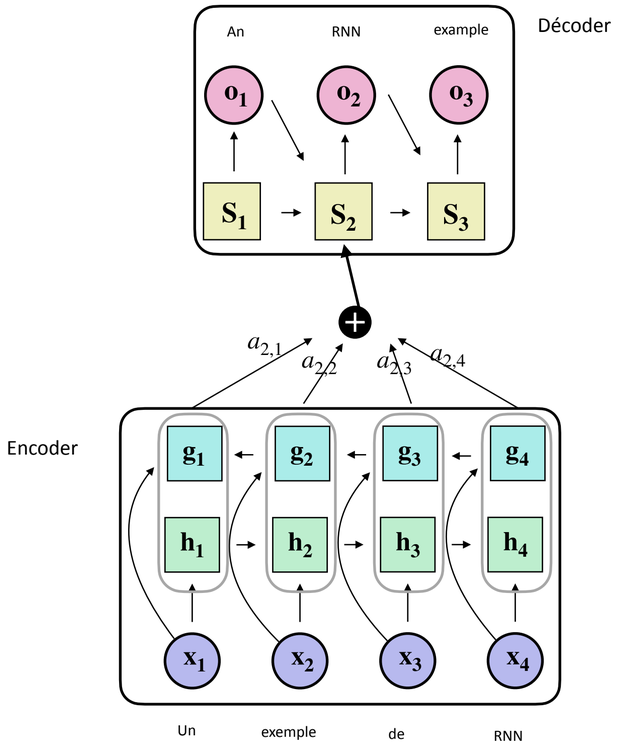
(Soft) Attention



a_{tj} : how much should decoder word t, "use" (attend to) on encoder word j

$$\sum_j a_{tj} = 1 \quad \forall t$$

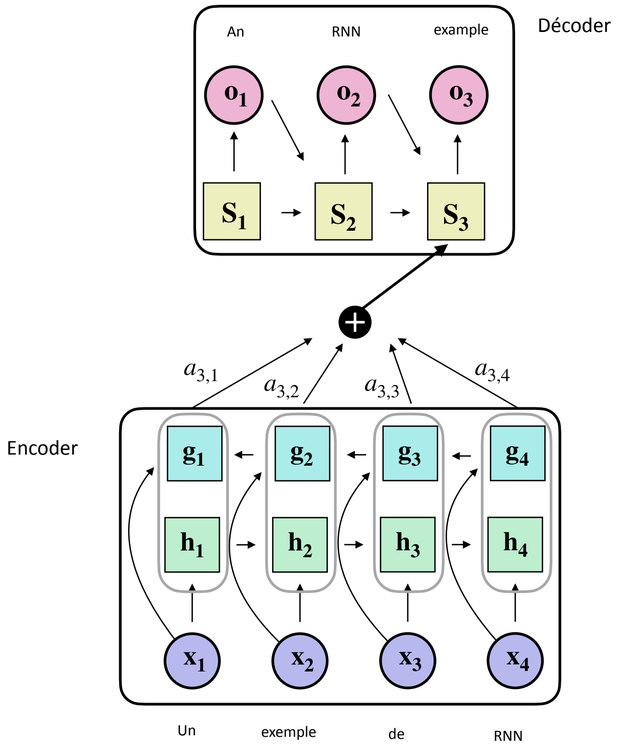
(Soft) Attention



a_{tj} : how much should decoder word t, "use" (attend to) on encoder word j

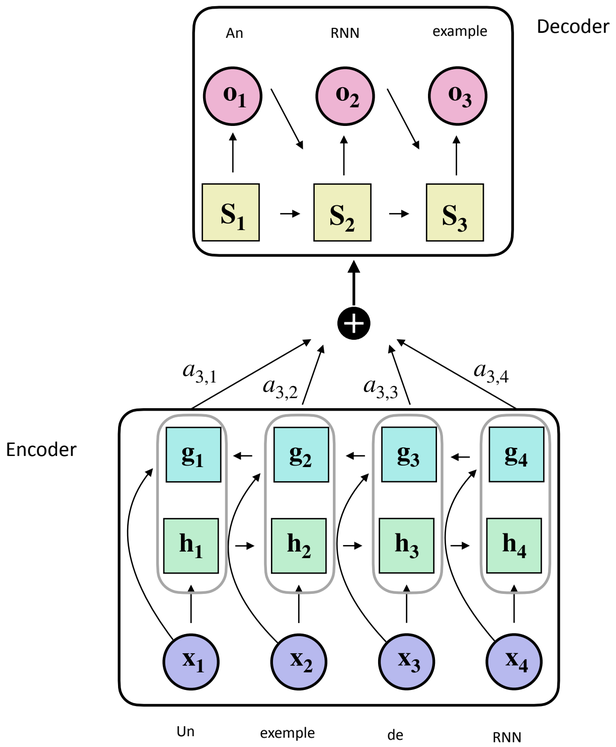
$$\sum_j a_{tj} = 1 \quad \forall t$$

(Soft) Attention



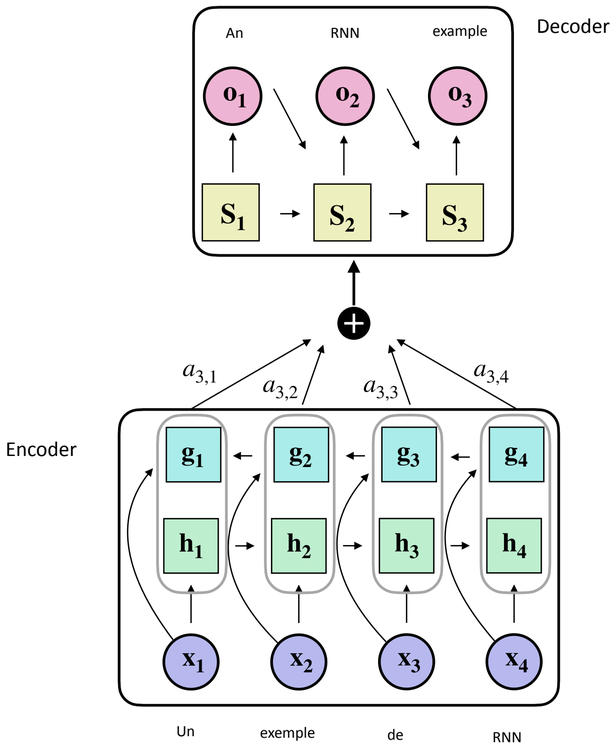
a_{tj} : how much should decoder word t, "use" (attend to) on encoder word j

$$\sum_j a_{tj} = 1 \quad \forall t$$



Advantages:

- *No more bottleneck*
- The decoder can consider different representations (information)
- The latent representation is now proportional to the length of the sequence.
- Shortcuts between the encoder and decoder
- Can model longer dependencies



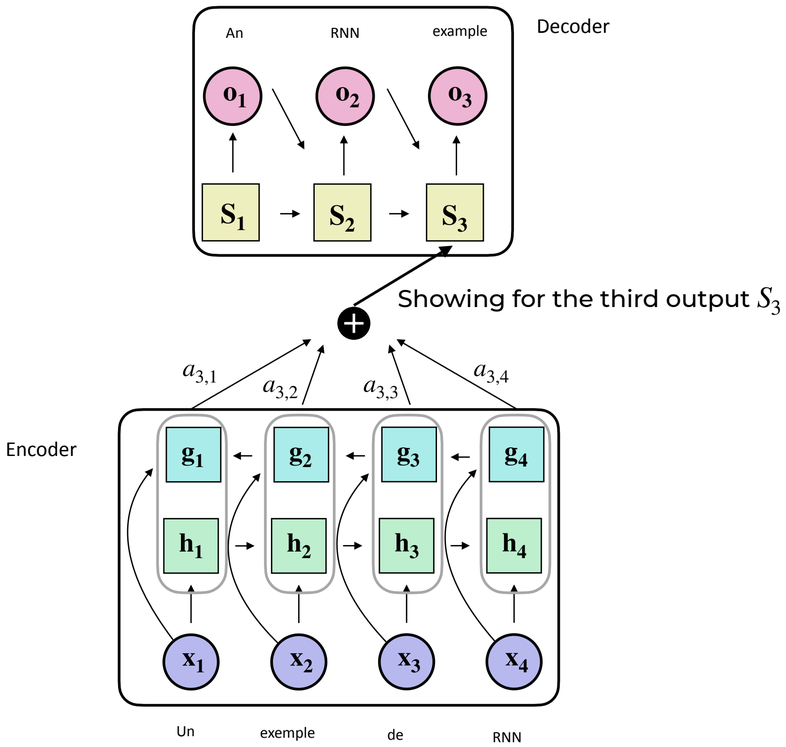
Mathematical details:

$$\mathbb{P}(\mathbf{o_t} \mid \mathbf{o_{1:t-1}}, \mathbf{x_{1:T}}) = f(\mathbf{s_t}, \mathbf{c_t}),$$

where

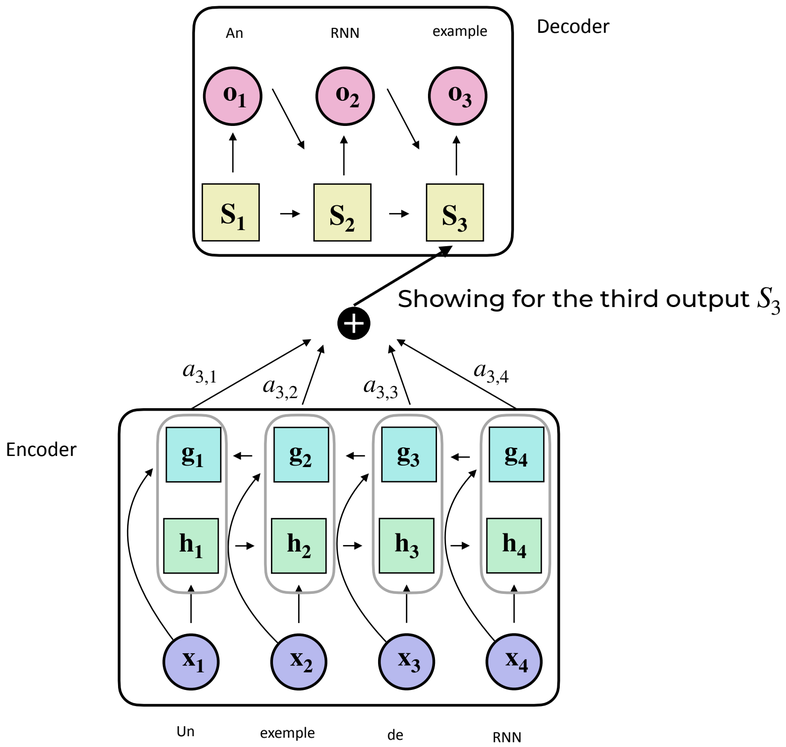
$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

Attention
- Timestep t (output)
- Hidden representation j (input)



Mathematical details:

The attention weights are obtained as follows:



Mathematical details:

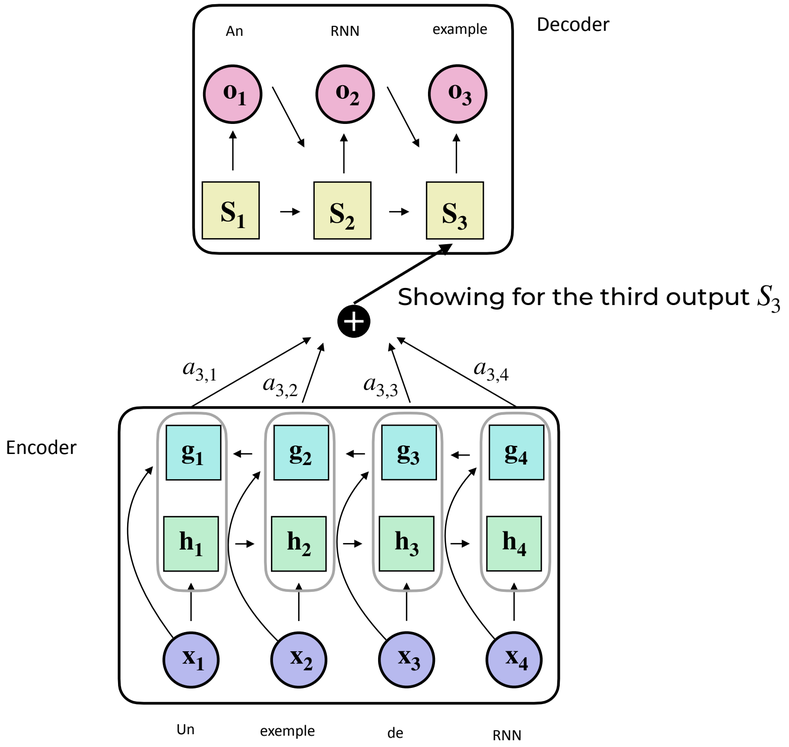
The attention weights are obtained as follows:

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}},$$

Where

$$\mathbf{e_{tj}} = \alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j})\,.$$

Function models the similarity between input and output representations



Mathematical details:

The attention weights are obtained as follows:

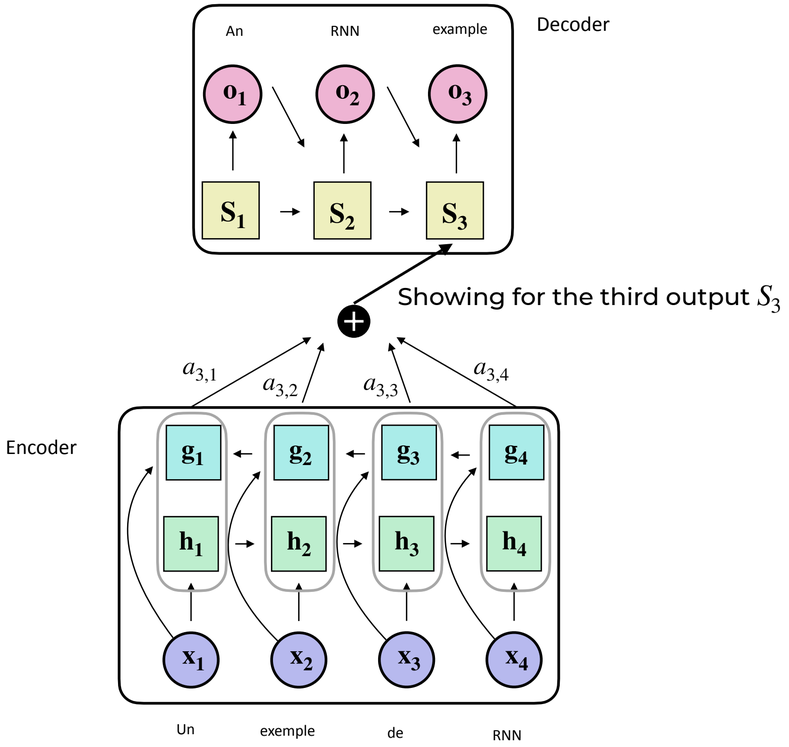
$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}},$$

Where

$$\mathbf{e_{tj}} = \alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j})\,.$$

Function models the similarity between input and output representations

The function $\alpha(\cdot)$ can be, for example, an MLP:



Mathematical details:

The attention weights are obtained as follows:

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}},$$

Where

$$\mathbf{e_{tj}} = \alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j}).$$

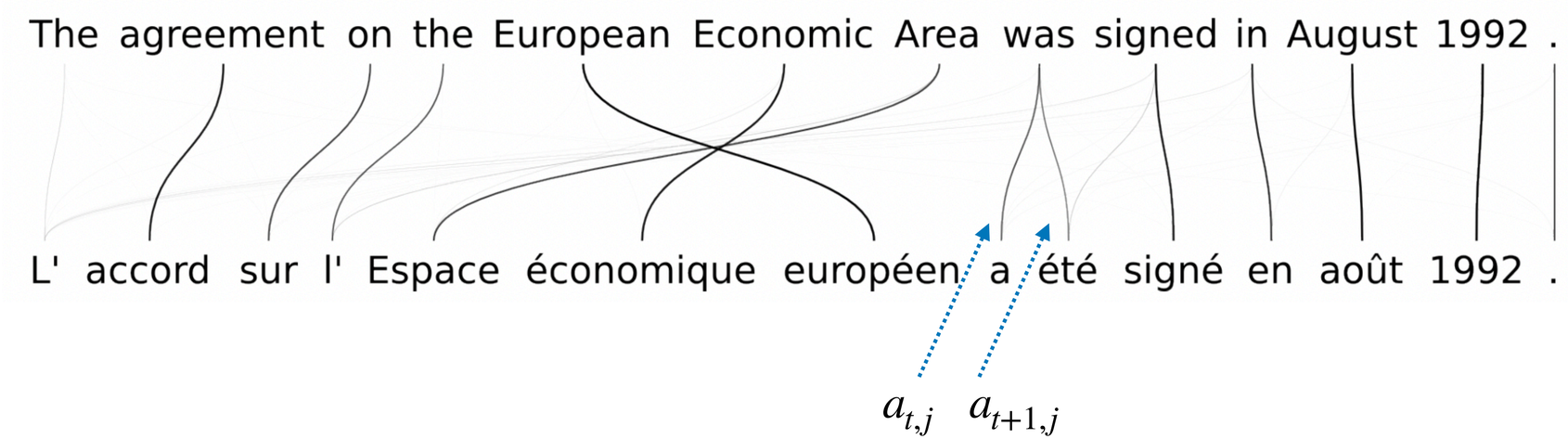
Function models the similarity between input and output representations

The function $\alpha(\cdot)$ can be, for example, an MLP:

$$\alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j}) = \mathbf{v_a^\top} \tanh\Big(\mathbf{W}_\alpha \mathbf{s_{t-1}} + \mathbf{U}_\alpha[\mathbf{h_j}, \mathbf{g_j}]\Big).$$

Visualizing attention

Translation task (EN -> FR)

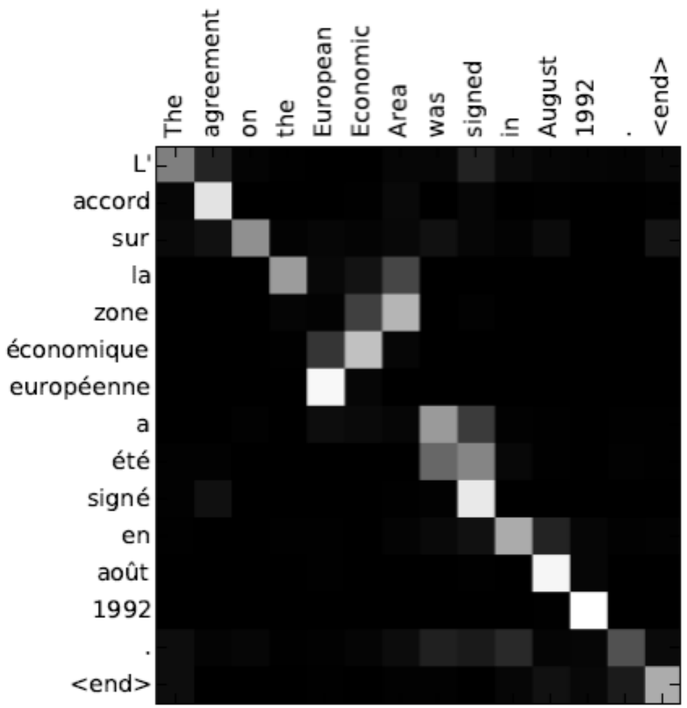


Visualizing attention

Translation task (EN -> FR)
(attention matrix)

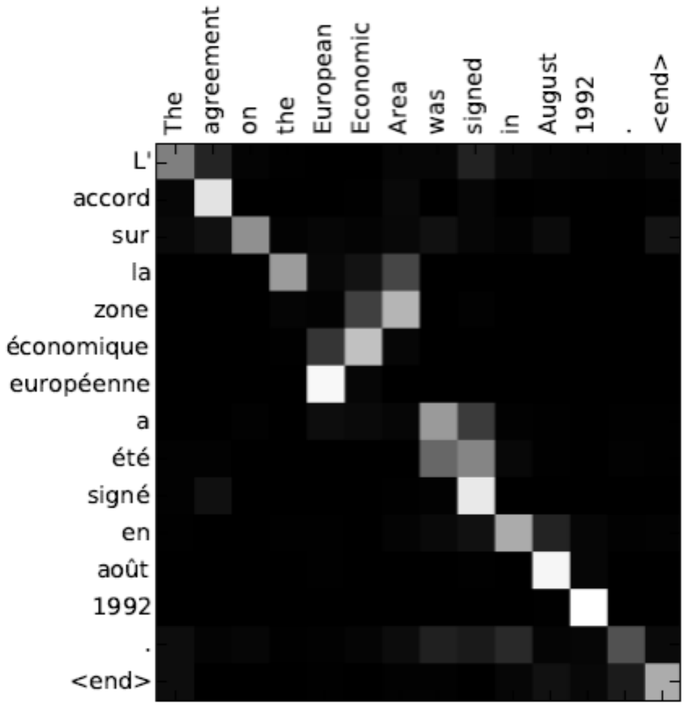
Visualizing attention

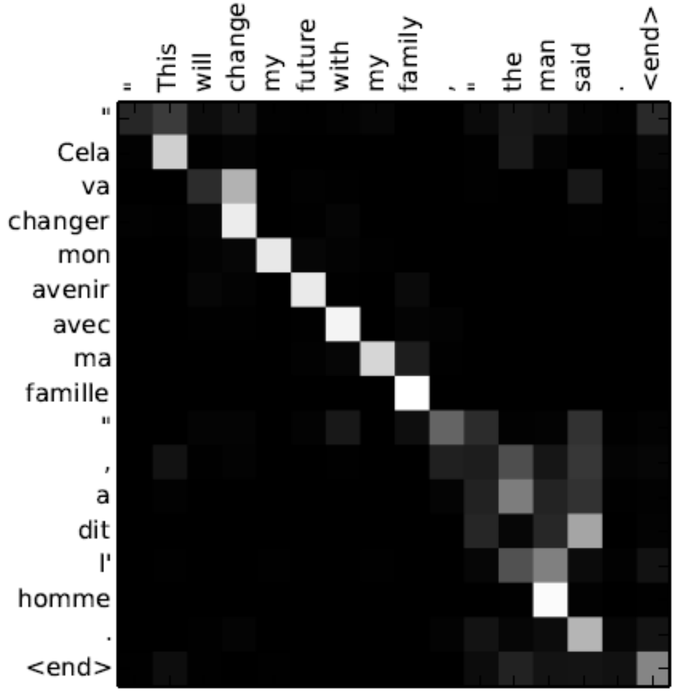
Translation task (EN -> FR)
(attention matrix)



Visualizing attention

Translation task (EN -> FR)
(attention matrix)

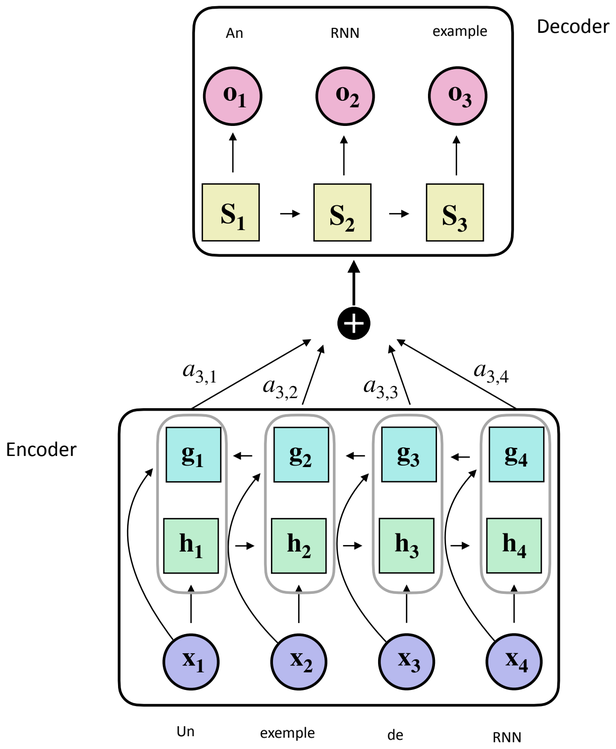




Soft Attention summary

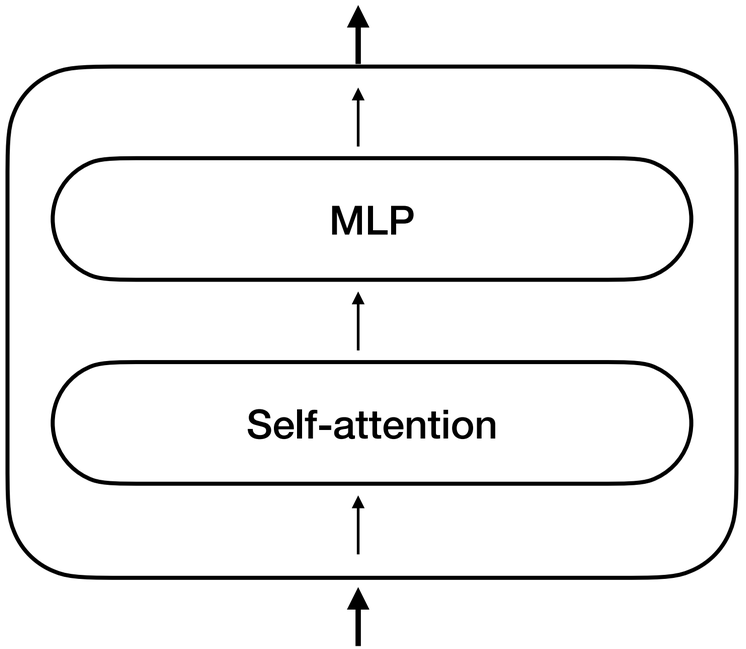
- **Dynamically weights the different contexts.**
- **Empirically: Works for length of up to ~100 time steps**

Toward Transformers



- A mix of sequential (h_i, g_j, S_t) and parallel (attention) processing
  - Expensive computationally
- Could a parallel architecture model sequential data?

Transformer Block



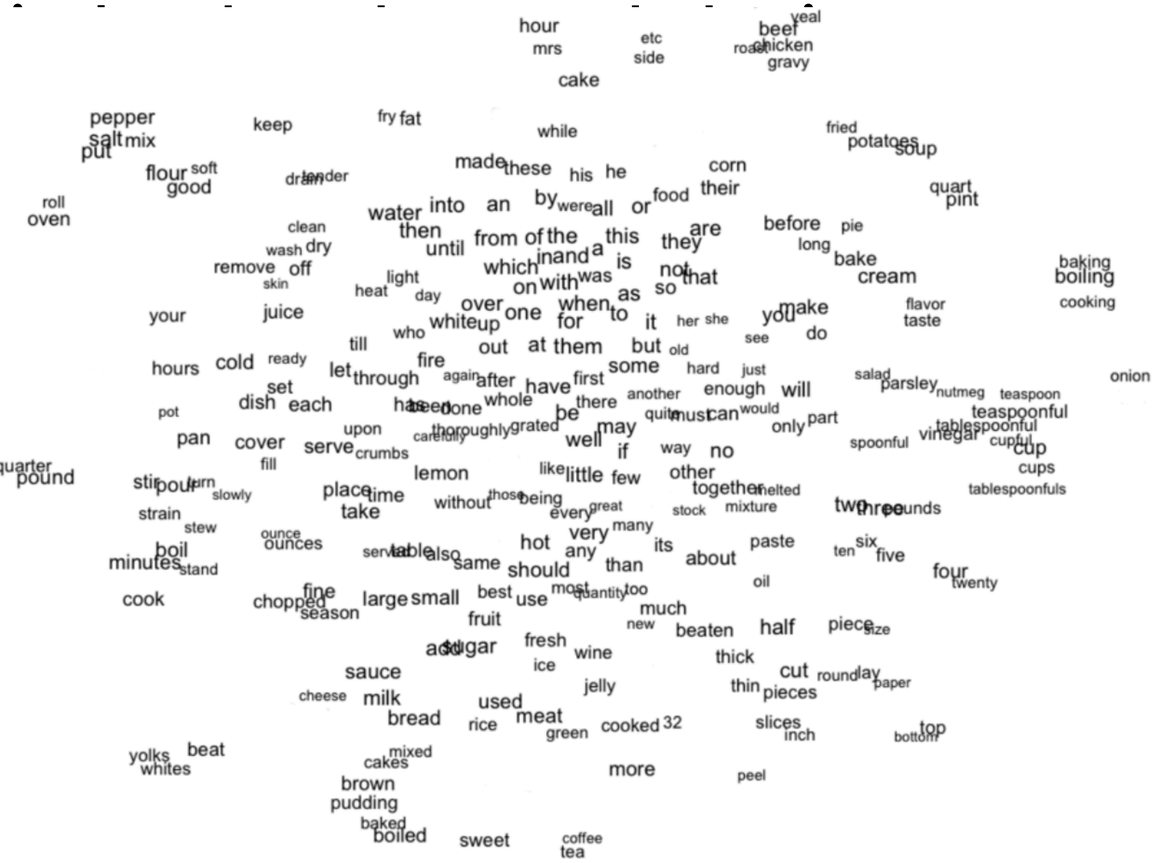
William Shakespeare (c. 23[a] April 1564 – 23 April 1616)[b] was an English playwright, poet and actor.
He is widely regarded as the greatest writer in the English language and the world's pre-eminent dramatist.[3][4][5]
He is often called England's national poet and the "Bard of Avon" (or simply "the Bard").
His extant works, including collaborations, consist of some 39 plays, 154 sonnets,
three long narrative poems and a few other verses, some of uncertain authorship.

Word embedding

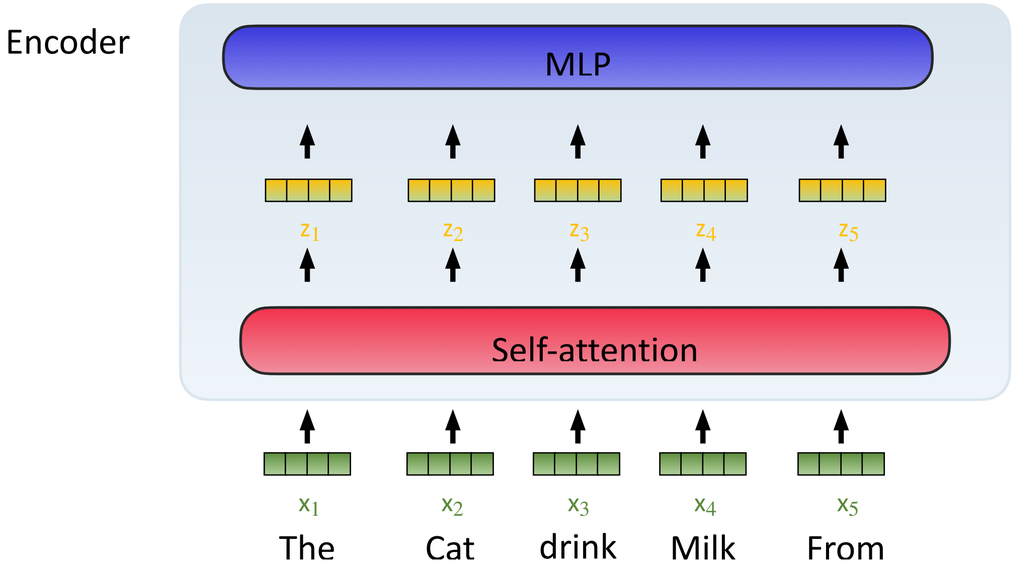
- Words are categorical and can be encoded using a 1-of-K encoding (i.e., dummies)
- This fails to capture the semantics of words between words (e.g., foot/feet/toe)
- In RNNs/Transformers/etc. words are represented using a vector (of size D)
  - The representation is learned
  - Smaller-dimensional representation (D << K)

Word embedding

- Words are categor
  of-K encoding (i.e.
- This fails to captur
  words (e.g., foot/fe
- In RNNs/Transforr
  using a vector (of
  - The representati
  - Smaller-dimensi



Self-attention

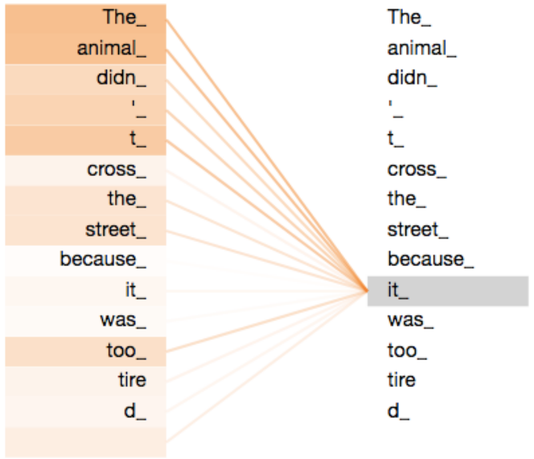


Self-attention

- The representation of each word (token) will be a combination of the representation of other words

Example:

'The animal didn't cross the street because it was too tired



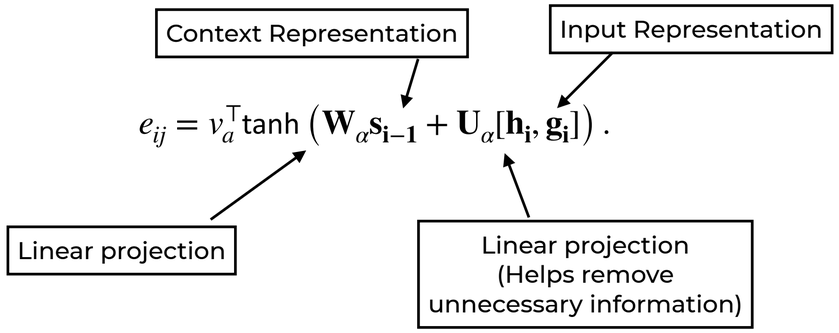
- The procedure is unsupervised (*self*)

Self-Attention

Mathematical Details

(Recall) Soft attention

$$\mathbf{a_{t,j}} = \left(\text{softmax}(\mathbf{e_{ij}})\right)_{\mathbf{j}},$$



Self-Attention

Mathematical Details

(Recall) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Context Representation

Input Representation

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Linear projection

Linear projection
(Helps remove
unnecessary information)

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

where

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i}\mathbf{W^k})(\mathbf{x_j}\mathbf{W^q})\Big)$$

Self-Attention

Mathematical Details

(Recall) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Context Representation

Input Representation

Linear projection

Linear projection
(Helps remove
unnecessary information)

Calculate its output

$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

where

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i W^k})(\mathbf{x_j W^q})\Big)$$

Self-Attention

Mathematical Details

(Recall) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Context Representation

Input Representation

Linear projection

Linear projection (Helps remove unnecessary information)

Calculate its output

$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

where

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i W^k})(\mathbf{x_j W^q})\Big)$$

x_i: input representation of word i (NxD)
z_j: output representation of word j (1xD)

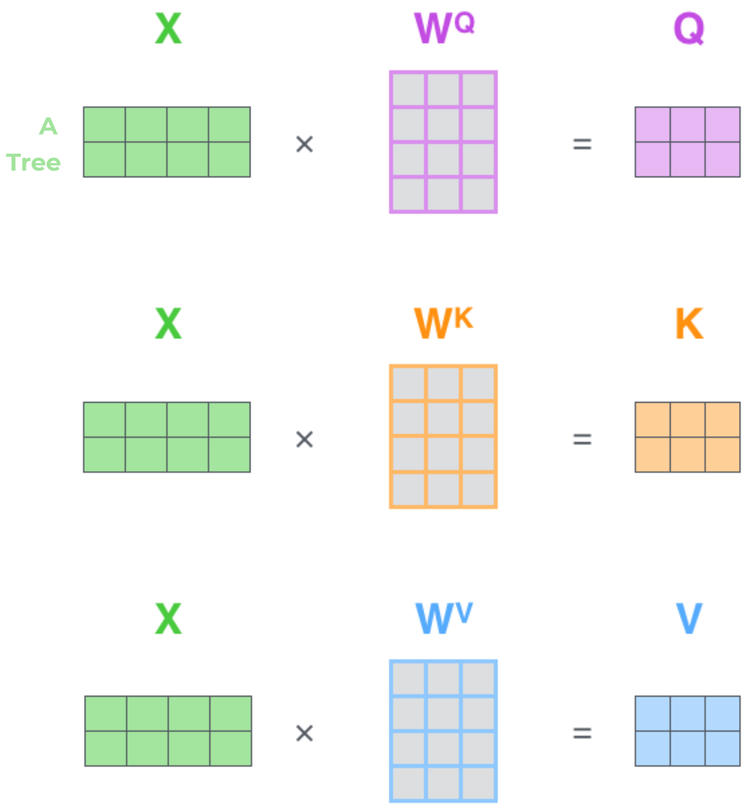
Calculate its output

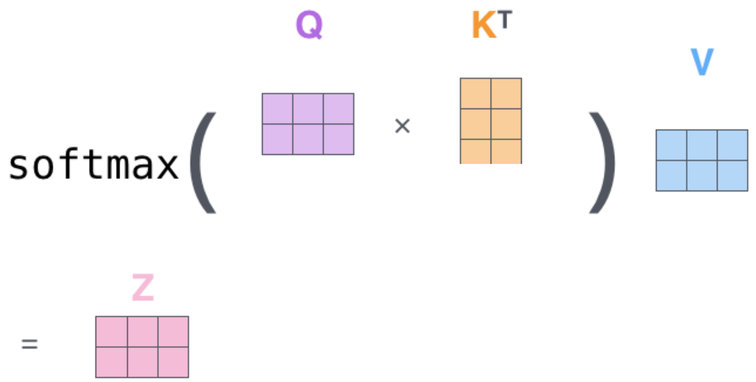
Often referred to as “Key-Query-Value” attention

$$\mathbf{z_j} = \underbrace{(\mathbf{x_i W^v})}_{\text{Value}}\ \mathbf{softmax}(\underbrace{(\mathbf{x_i W^k})}_{\text{Key}}\ \underbrace{(\mathbf{x_j W^q})}_{\text{Query}})$$

Self-Attention

Visual representation
N=2, D=4





*: in practice the value of attention pre-softmax (QK^T) is often normalized by $\sqrt{D}$ for optimisation stability

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

Multi-Head Self-Attention (MHSA)

- Attention captures the similarity between two words
- One may wish to capture several different similarities (e.g., whale and humans are both mammals, whales and sharks live in the water)
- You can learn the parameters for multiple attention mechanisms. Each mechanism is called an attention head:

$$\mathbf{x'_j} = [\mathbf{z_j^1 z_j^2 \cdots z_j^H}]\mathbf{W^0}$$

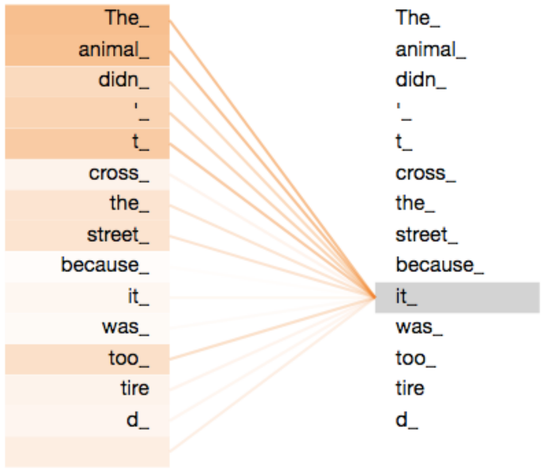
x'_j : the representation of word j after self-attention (D)
z_j^h: the representation of word j from attention head h (D')
H: the number of heads
W^0 : weight matrix (D'H x D)

Multi-Head Self-Attention (MHSA)

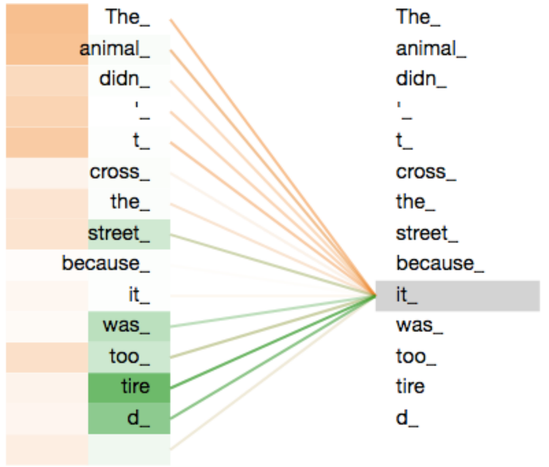
Example:

'The animal didn't cross the street because it was too tired'

One head



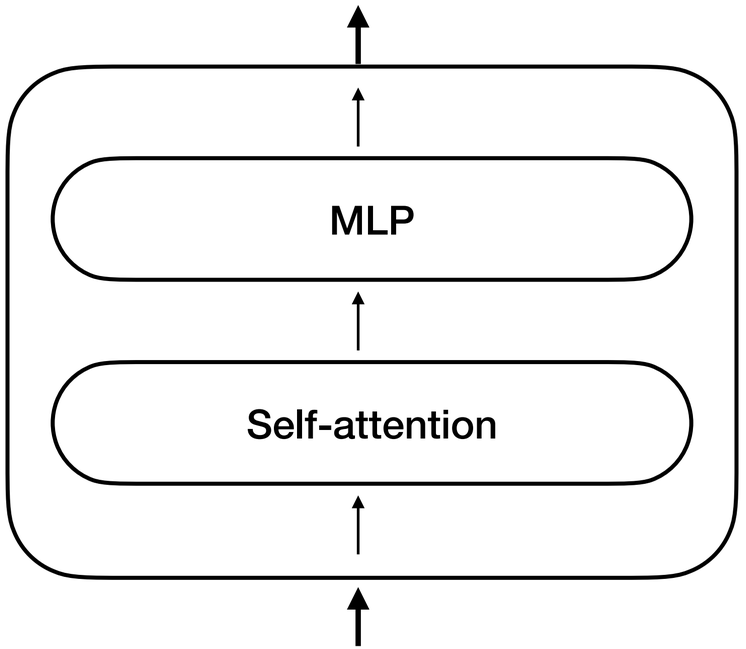
Two heads



Demo

- https://poloclub.github.io/transformer-explainer/
  - Shows attention
  - Sub-word tokens

Transformer Block



Operates across features (D)

(Mostly) Operates across tokens (N)

William Shakespeare (c. 23[a] April 1564 – 23 April 1616)[b] was an English playwright, poet and actor.
He is widely regarded as the greatest writer in the English language and the world's pre-eminent dramatist.[3][4][5]
He is often called England's national poet and the "Bard of Avon" (or simply "the Bard").
His extant works, including collaborations, consist of some 39 plays, 154 sonnets,
three long narrative poems and a few other verses, some of uncertain authorship.

Transformers… transform?

- Transformer blocks rely on attention heads followed by an MLP
- How can we use them for language tasks (e.g., translation)?
- Transformer blocks can be combined and refined
  - Transformer blocks can be used as *encoders* & *decoders*
  - For decoding, a few changes are required

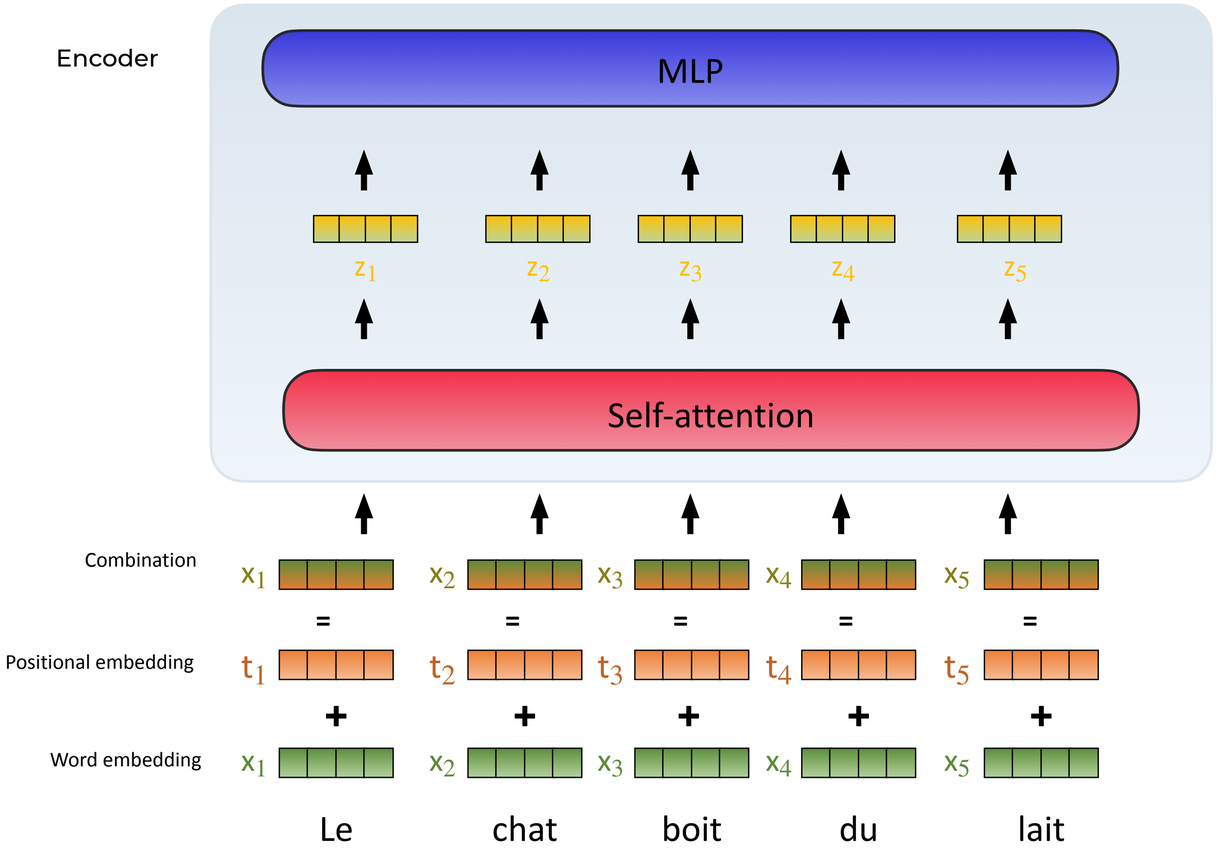
A few other characteristics

- Word embeddings
  - Words are categorical and can be encoded using a 1-of-K encoding (i.e., dummies)
  - This fails to capture the semantics of words between words (e.g., foot/feet/toe)
  - In RNNs/Transformers/etc. words are represented using a vector (of size D)

A few other characteristics

- Positional embedding
  - Attention looses the position of words into account
    - In many tasks, position is essential
      - “cat eats plant” is different from “plant eats cat”
  - As a remedy, transformers also learn positional representations
    - Vectors that are specific to the position (can be absolute or relative)

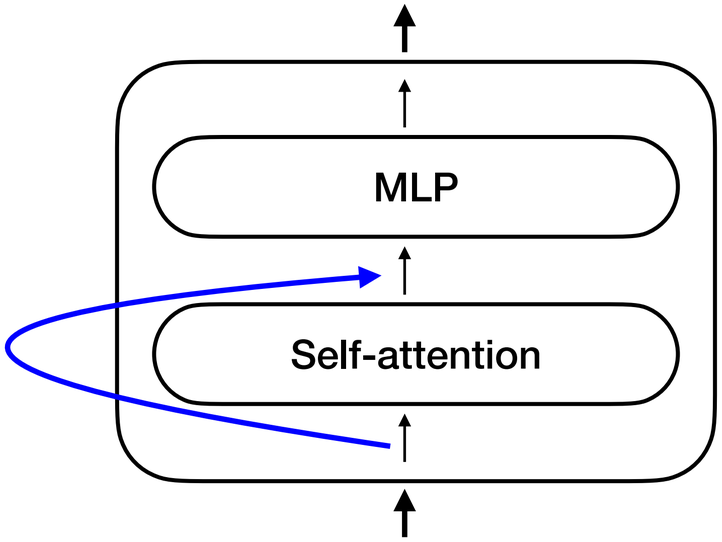
Word + Positional embeddings



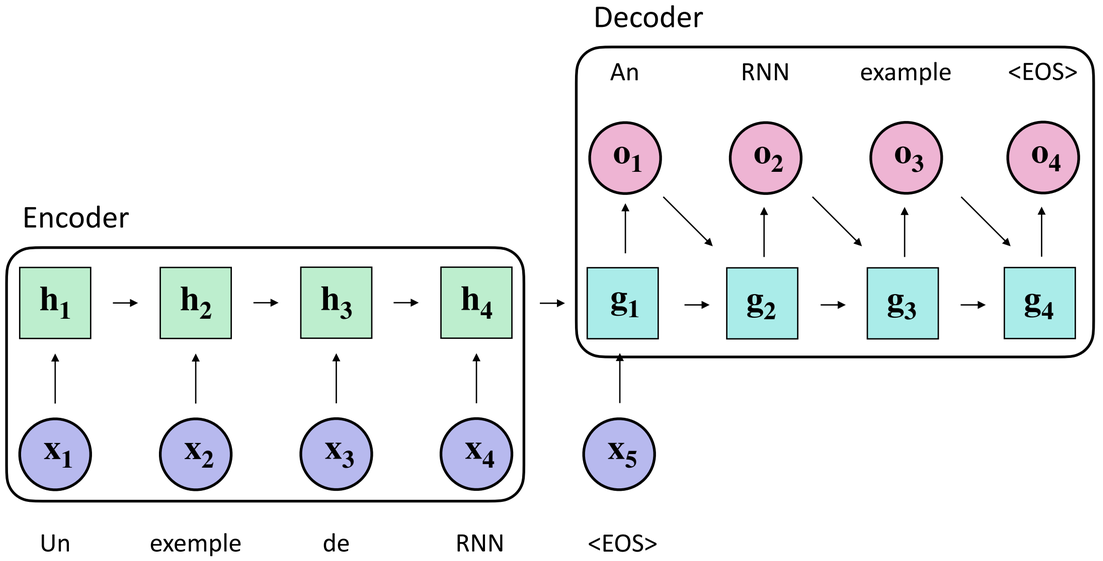
31

A few other characteristics

- Normalization
  - Layer normalization is often applied. It helps to learn by normalizing each token's dimensions to have 0-mean and 1 standard deviation.
- Residual connections
  - $z_i = \text{Self-Attention}(X) + x_i$
  - Biases learned functions to be "simple", helps to learn
  - Can be used for self-attention & MLP



What about decoding?



What about decoding?

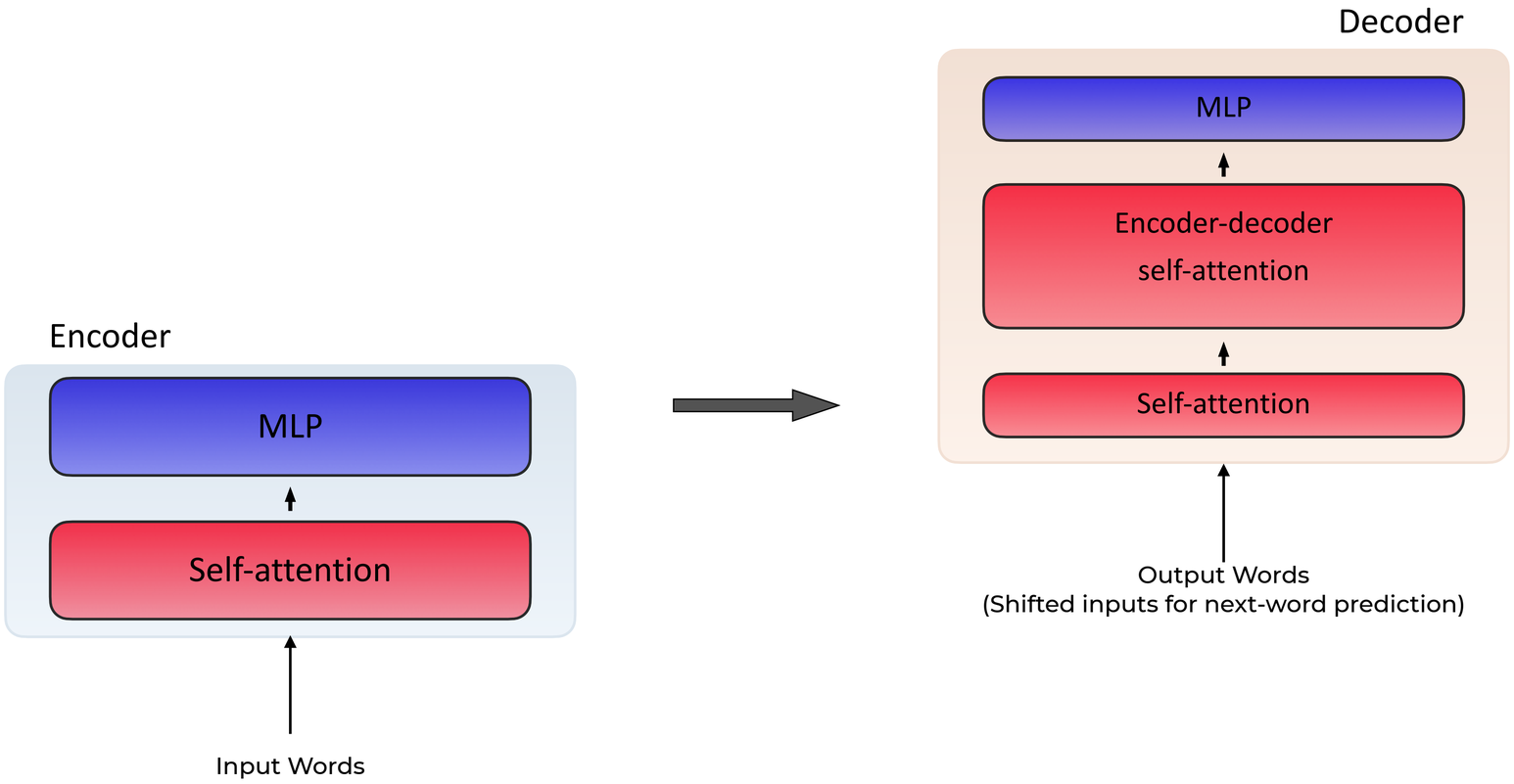
- Transformers transform word representations
  - Decoding requires using these words representations to obtain the following word
  - Add a “softmax”-layer at the end:

$$P(\mathbf{o_n} \mid \mathbf{o_{1:n-1}}) = \frac{\exp(\mathbf{g_w^\top x_{n-1}})}{\sum_{\mathbf{w}}^{\mathbf{W}} \exp(\mathbf{g_w^\top x_{n-1}})}$$

g_w: parameters
W: number of words in the vocabulary

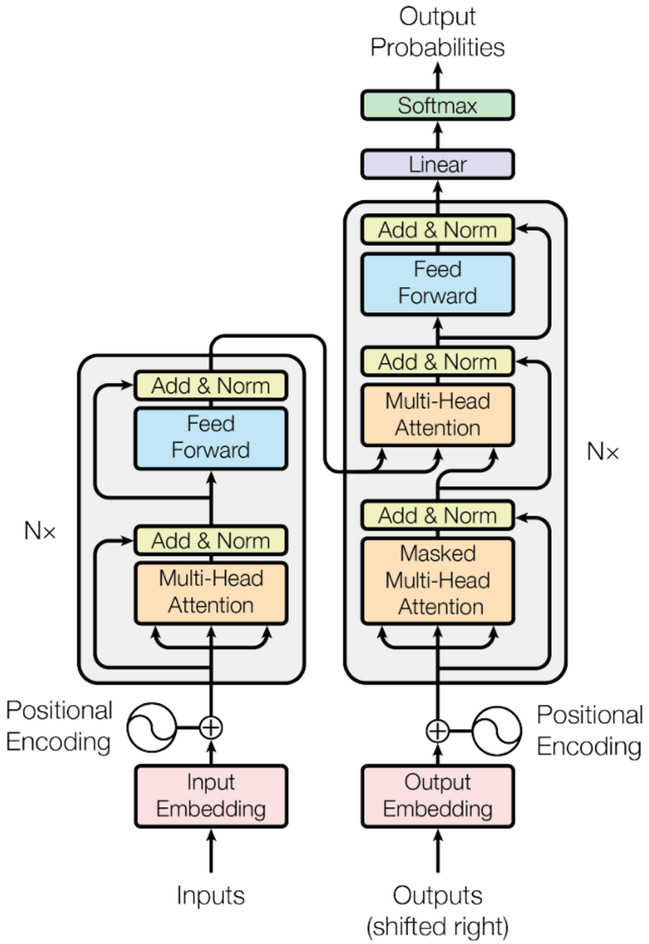
  - You can only attend to previous words
    - Attention matrix is constrained to be (upper) triangular

Encoder-Decoder Transformer



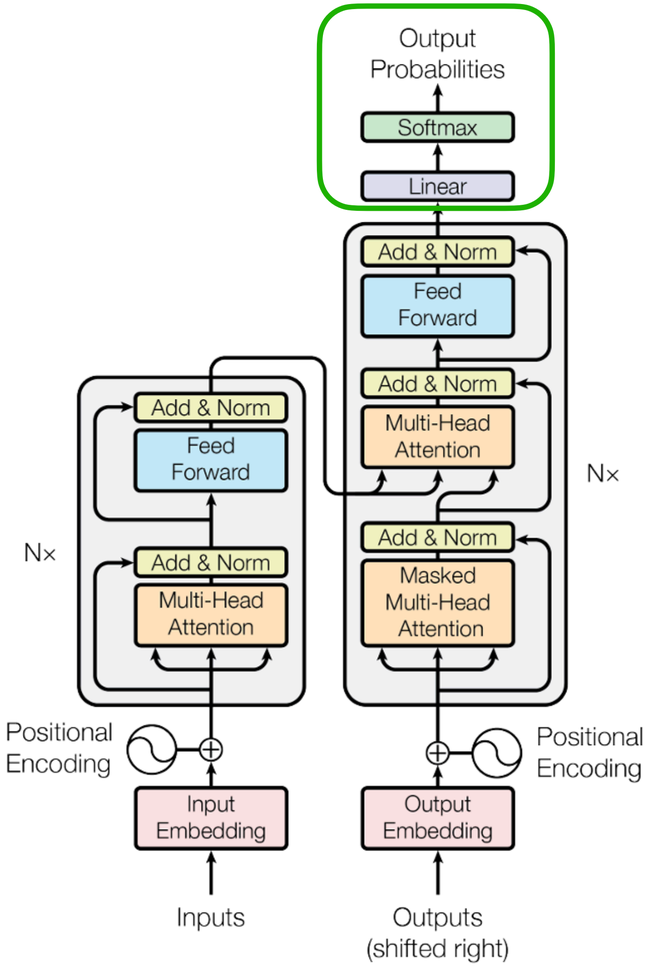
Putting it all together

Putting it all together



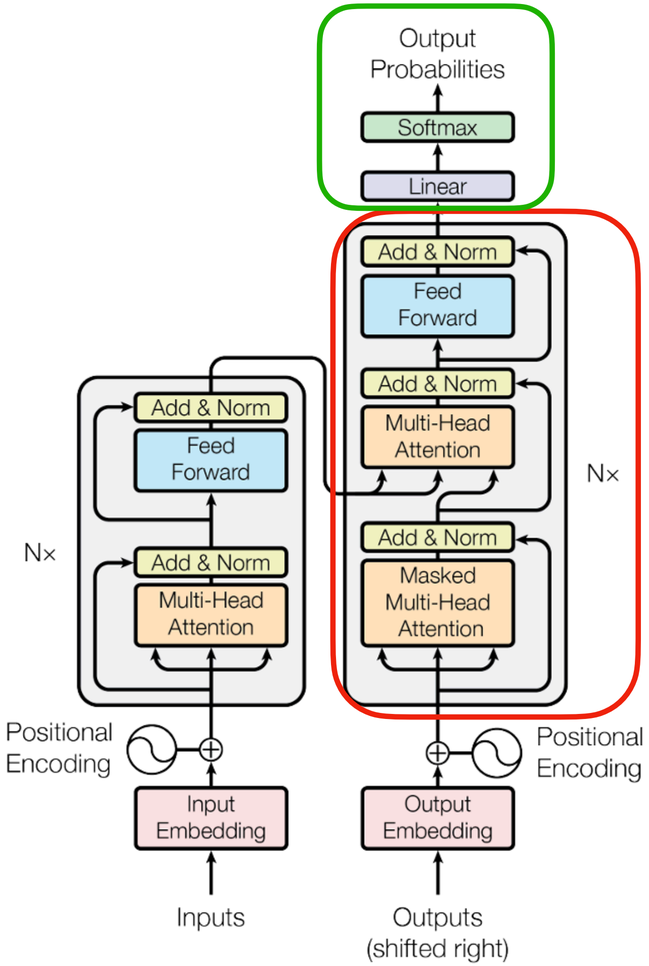
- Self-Attention
- Positional embedding
- Encoder
- Decoder
- Softmax-layer
- Multiple transformer blocks (Nx)

Putting it all together



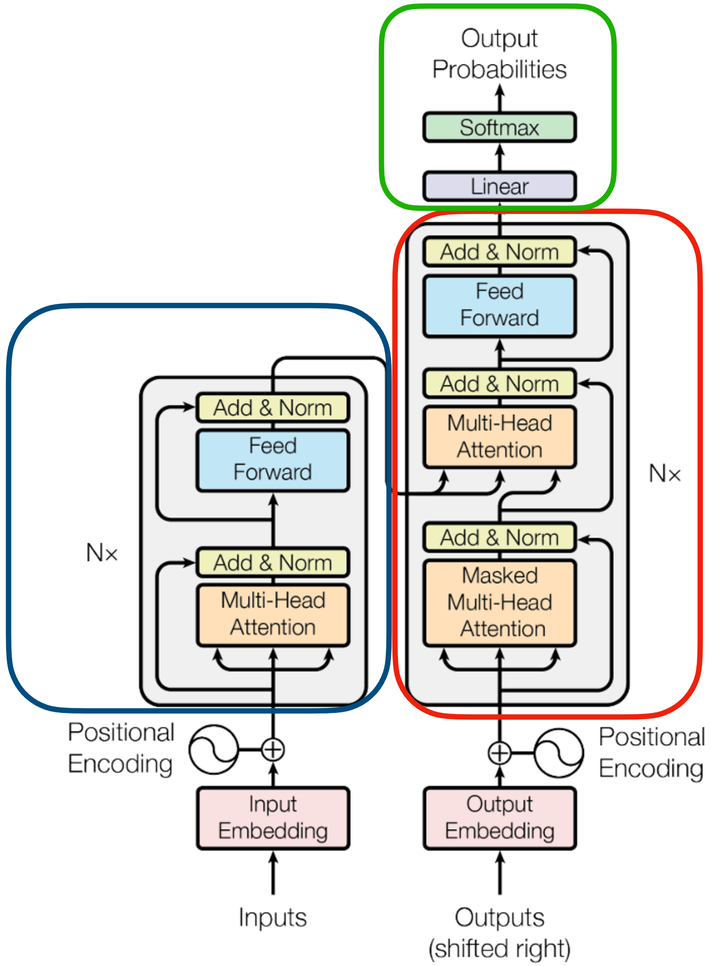
- Self-Attention
- Positional embedding
- Encoder
- Decoder
- Softmax-layer
- Multiple transformer blocks (Nx)

Putting it all together



- Self-Attention
- Positional embedding
- Encoder
- Decoder
- Softmax-layer
- Multiple transformer blocks (Nx)

Putting it all together



- Self-Attention
- Positional embedding
- Encoder
- Decoder
- Softmax-layer
- Multiple transformer blocks (Nx)

Objective

- Often in two (or more) stages:
  - **First stage**
    - Next-word prediction
    - Random-word prediction (mask some words in the input/output)
    - Other tasks:
      - Next-sentence prediction
  - **Second stage**
    - Preference and/or downstream task fine-tuning

One of many transformers

- Partial list: https://huggingface.co/docs/transformers/en/index
- Encoder-only models:
  - BERT (RoBERTa), ALBERT
- Encoder-Decoder models:
  - BART
- Decoder-only models:
  - Generative pre-trained transformer (GPT), BLOOM (for code), Llama
- *Most of the above transformers refer to a system that was trained not only an architecture. Many of these models now have tens of billions of parameters (Llama-7B -> 7 billion parameters)
- Also used for image modelling, audio, multi-modalities, etc.

Summary

- Self-attention is the key ingredient
  - Efficient and provides good results
  - Recent LLMs can attend to very long contexts (some refinements of attention)
  - Enables training transformers on much larger-scale datasets (internet size)
- Transformers has become the standard model often surpassing the performance of RNNs/CNNs when trained on enough data

References

- https://jalammar.github.io/illustrated-transformer/
- An Introduction to Transformers, Richard E. Turner, 2024, https://arxiv.org/abs/2304.10557
- Attention Is All You Need, Vaswani et al., 2017, http://arxiv.org/abs/1706.03762