Attention Mechanisms

Marcos V. Treviso

Deep Structured Learning

Fall 2020

December 2, 2020

Acknowledgments

Neural Attention Mechanisms by Ben Peters
Learning with Sparse Latent Structure by Vlad Niculae
Seq2Seq and Attention by Lena Voita
The elephant in the interpretability room by Jasmijn Bastings

The illustrated transformer
http://jalammar.github.io/illustrated-transformer/

The annotated transformer
http://nlp.seas.harvard.edu/2018/04/03/attention.html

Łukasz Kaiser’s presentation
https://www.youtube.com/watch?v=rBCqOTEfxvg

Summary

Quick recap on RNN-based seq2seq models

Attention and its different flavors

dense • sparse • soft • hard • structured

Self-attention networks

The Transformer

Attention interpretability

Can we consider attention maps as explanation?

Why attention?

- Attention is a recent and important component to the success of modern neural networks

- We want neural nets that **automatically weigh** relevance of the input and **use these weights** to perform a task

- Main advantages:
    - performance gain 🎯
    - none or few parameters ☁️
    - fast (easy to parallelize) ⚡
    - drop-in implementation ⚓
    - tool for "interpreting" predictions 🔍

Example

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

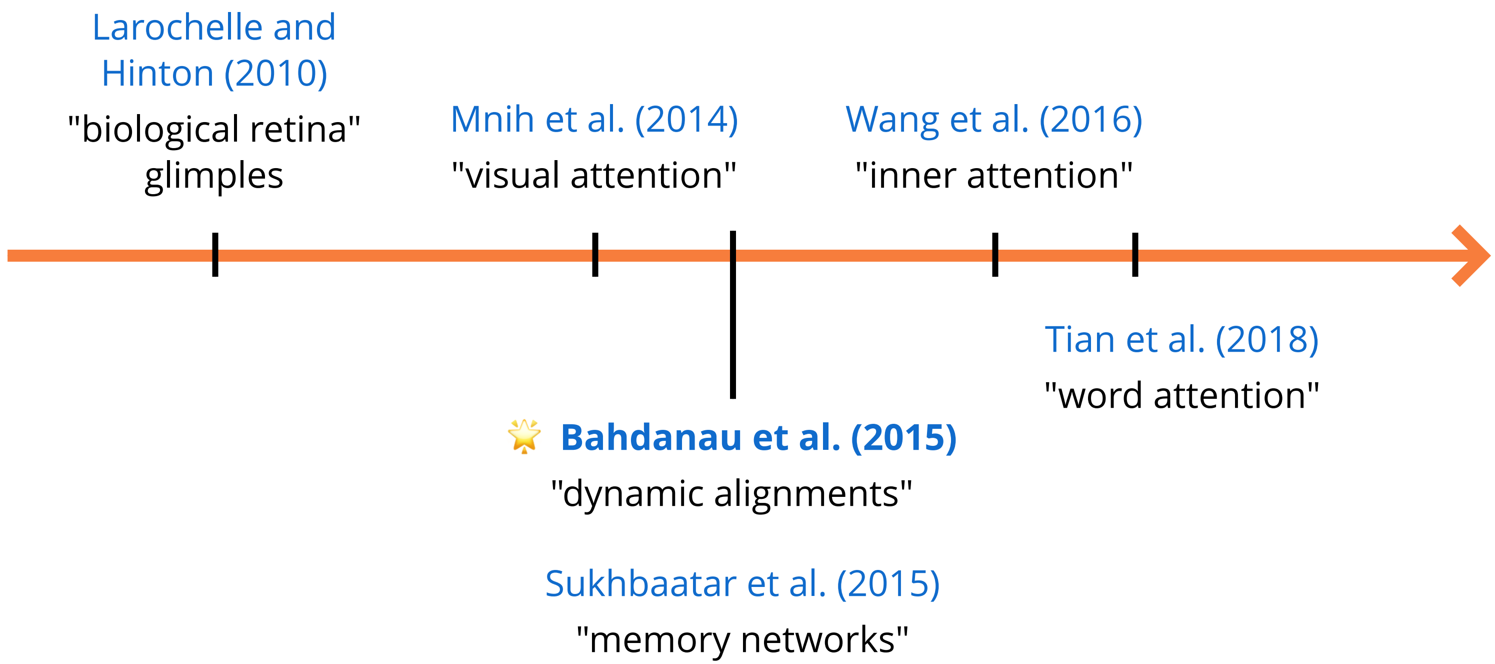
(Bao et al., 2018)

Example

(Martins et al., 2020)

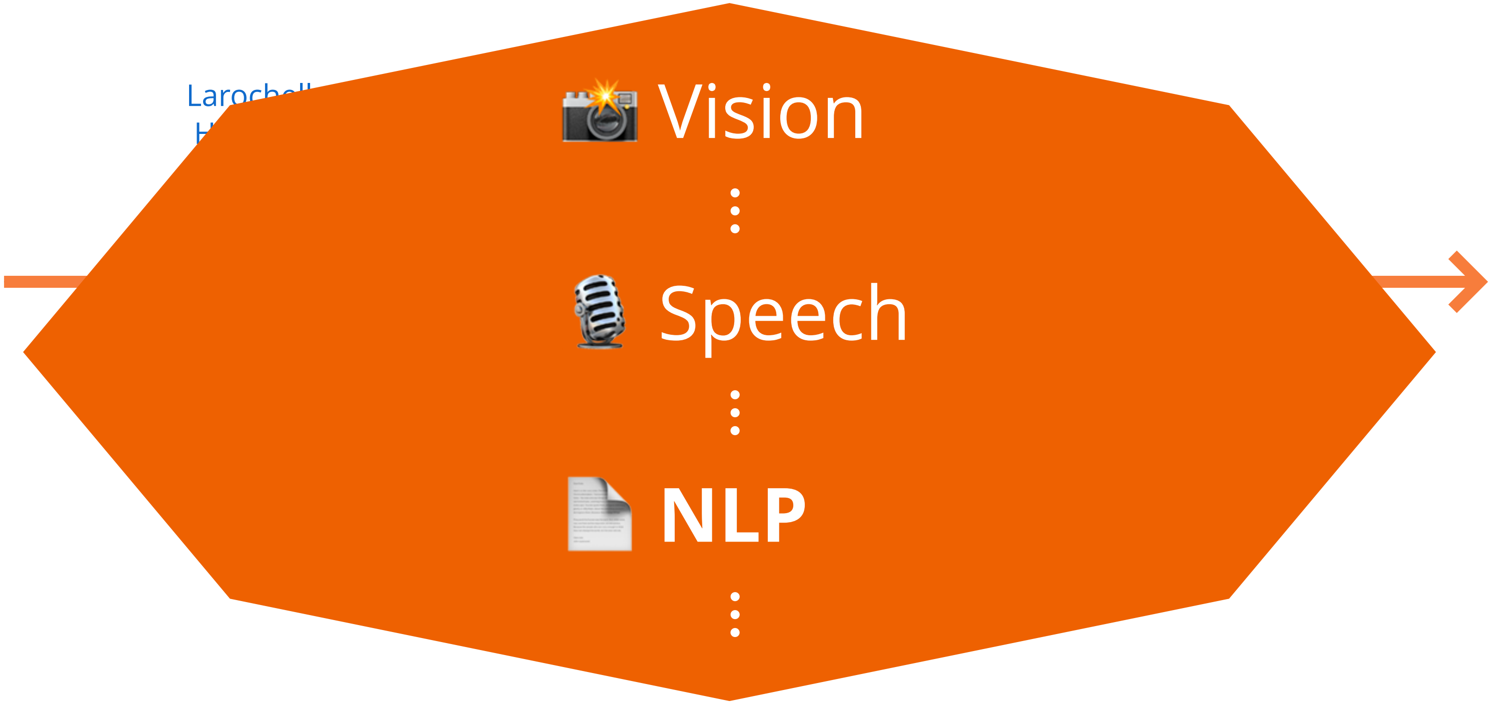
Brief history

- "first" introduced in NLP for Machine Translation by Bahdanau et al. (2015)



Brief history

- "first" introduced in NLP for Machine Translation by Bahdanau et al. (2015)

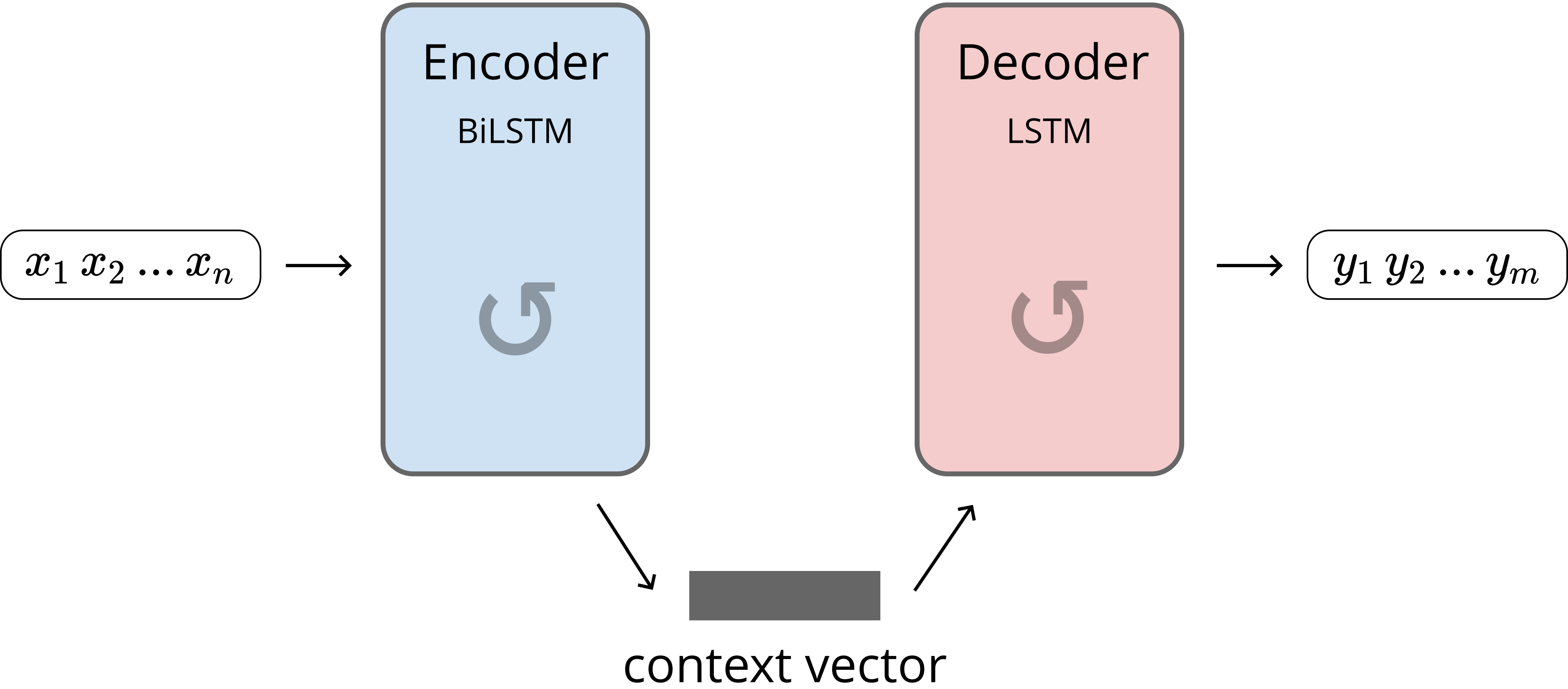


Attention in NLP

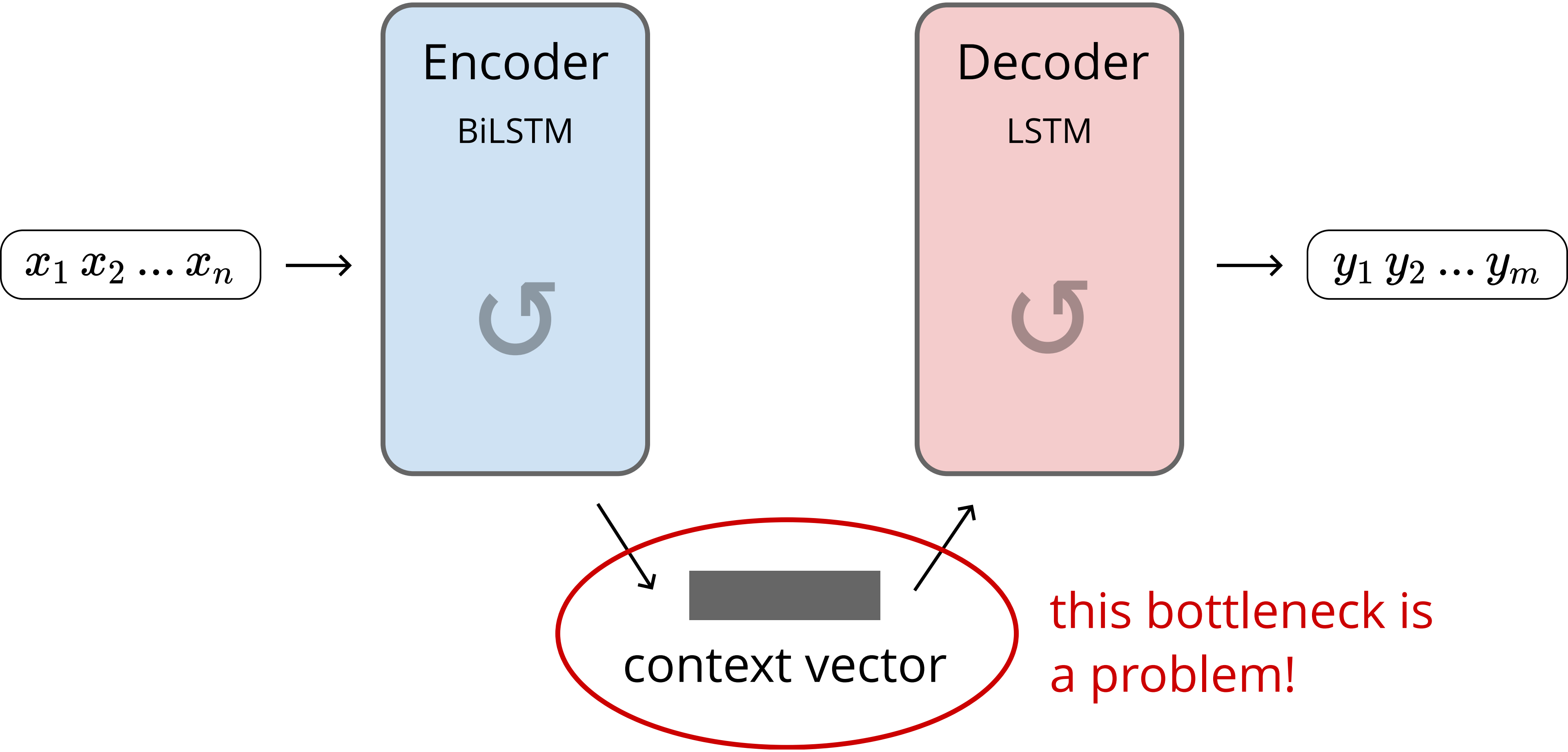
Task Addressed	Related Works
Machine Translation	[2, 6, 8, 29–48, 48–50]
Translation Quality Estimation	[51]
Text Classification	[7, 8, 10, 11, 52, 53]
Abusive content detection	[54]
Text Summarization	[41, 55–58]
Language Modelling	[59–61]
Question Answering	[10, 47, 59, 62–75]
Question Answering over Knowledge Base	[76]
Morphology	
Pun Recognition	[77]
Multimodal Tasks	[78]
Image Captioning	[16, 79]
Visual Question Answering	[80–82]
Task-oriented Language Grounding	[83]
Information Extraction	
Coreference Resolution	[84, 85]
Named Entity Recognition	[51, 86]
Optical Character Recognition Correction	[87]
Semantic	
Entity Disambiguation	[88]
Natural Language Inference	[8, 10, 47, 89–96]
Semantic Relatedness	[93]
Semantic Role Labelling	[97, 98]
Sentence Similarity	[96]
Textual Entailment	[75, 99, 100]
Word Sense Disambiguation	[101]
Syntax	
Constituency Parsing	[102, 103]
Dependency Parsing	[51, 104, 105]
Sentiment Analysis	[1, 7, 93, 95, 100, 106–120]
Agreement/Disagreement Identification	[121]
Argumentation Mining	[57, 122–125]
Emoji prediction	[126]
Emotion Cause Analysis	[127, 128]
Emotion Classification	[115]

(Galassi et al., 2020)

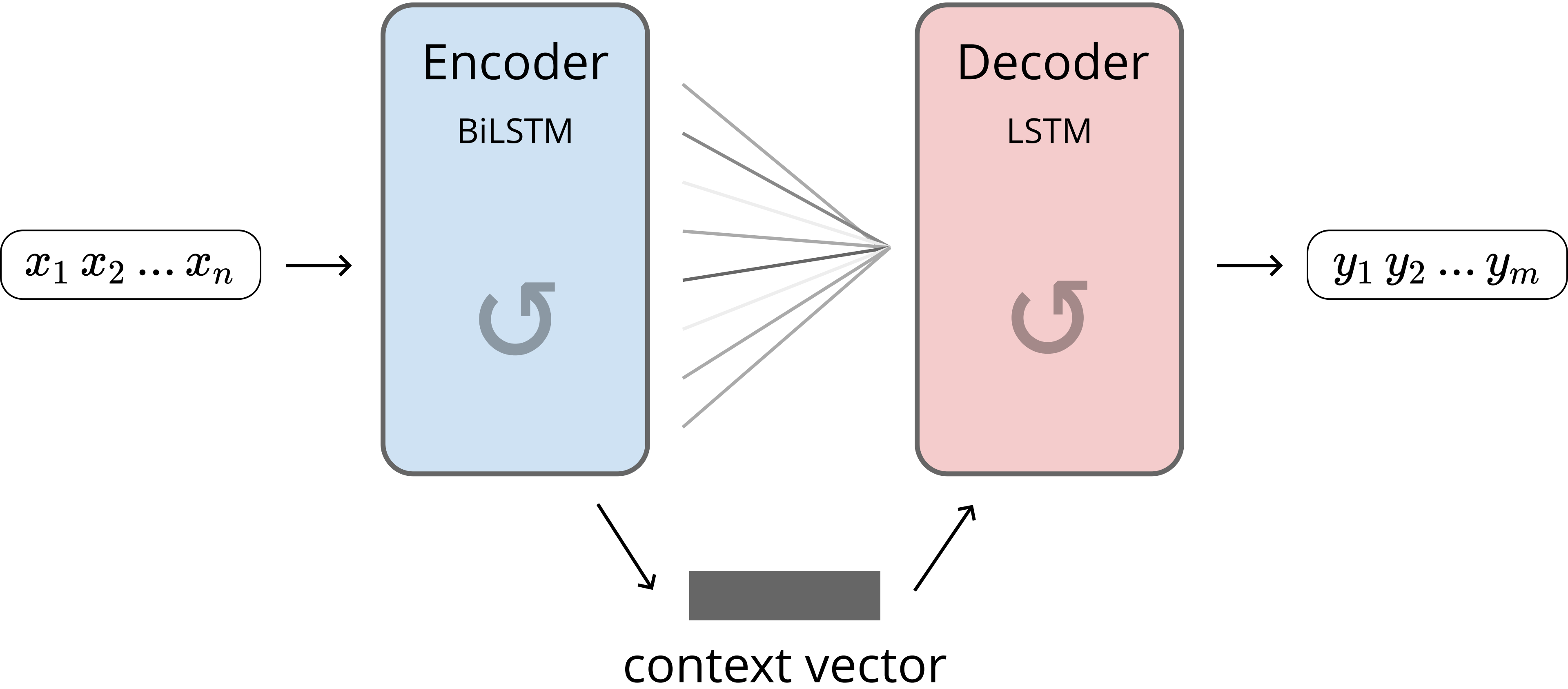
RNN-based seq2seq



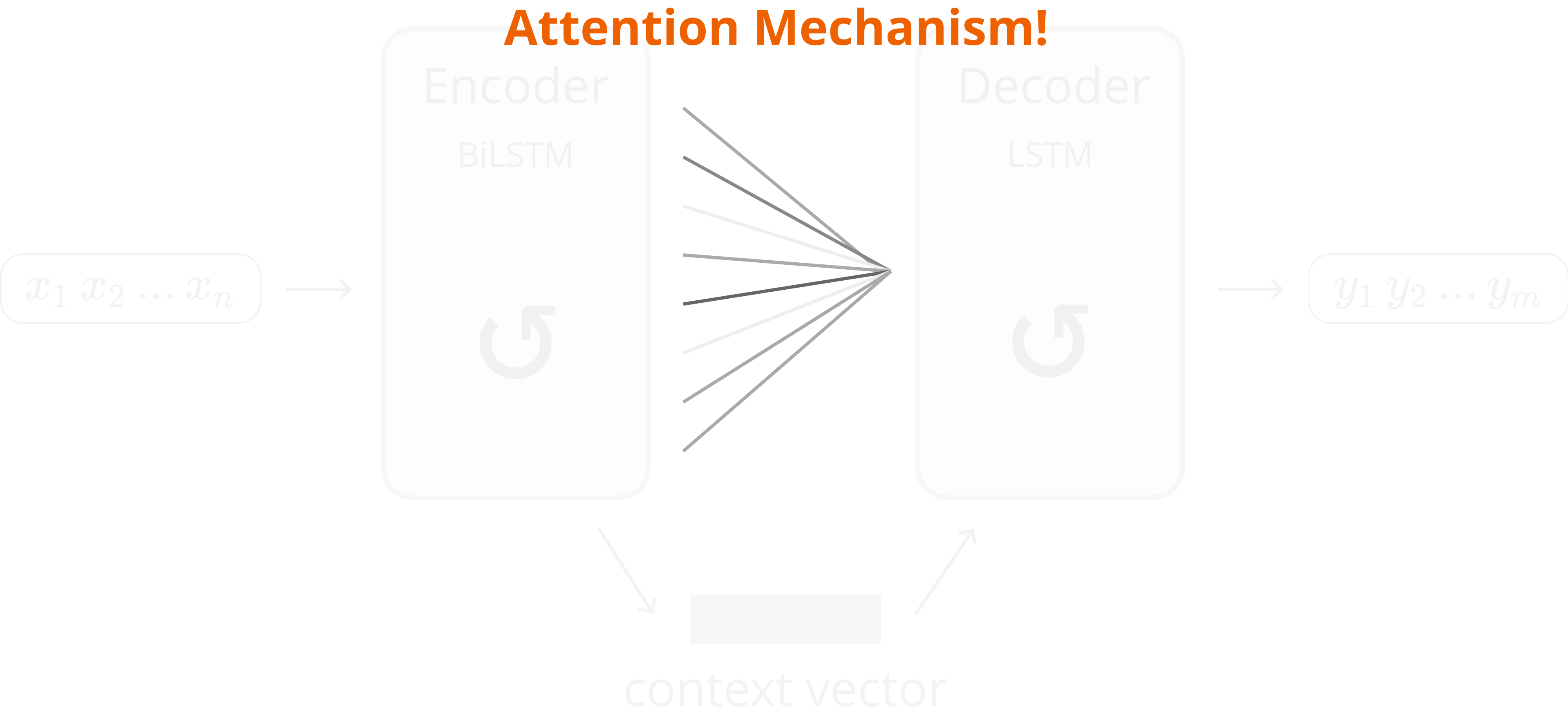
RNN-based seq2seq



RNN-based seq2seq

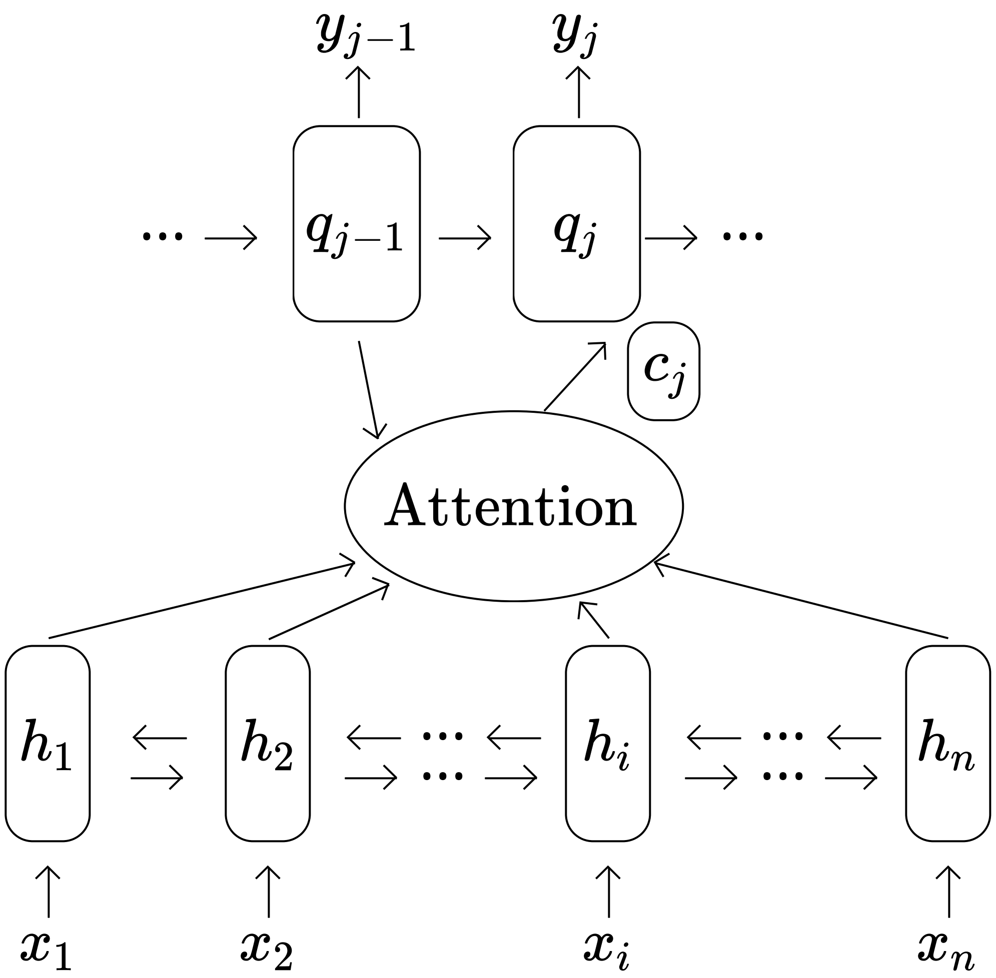


RNN-based seq2seq



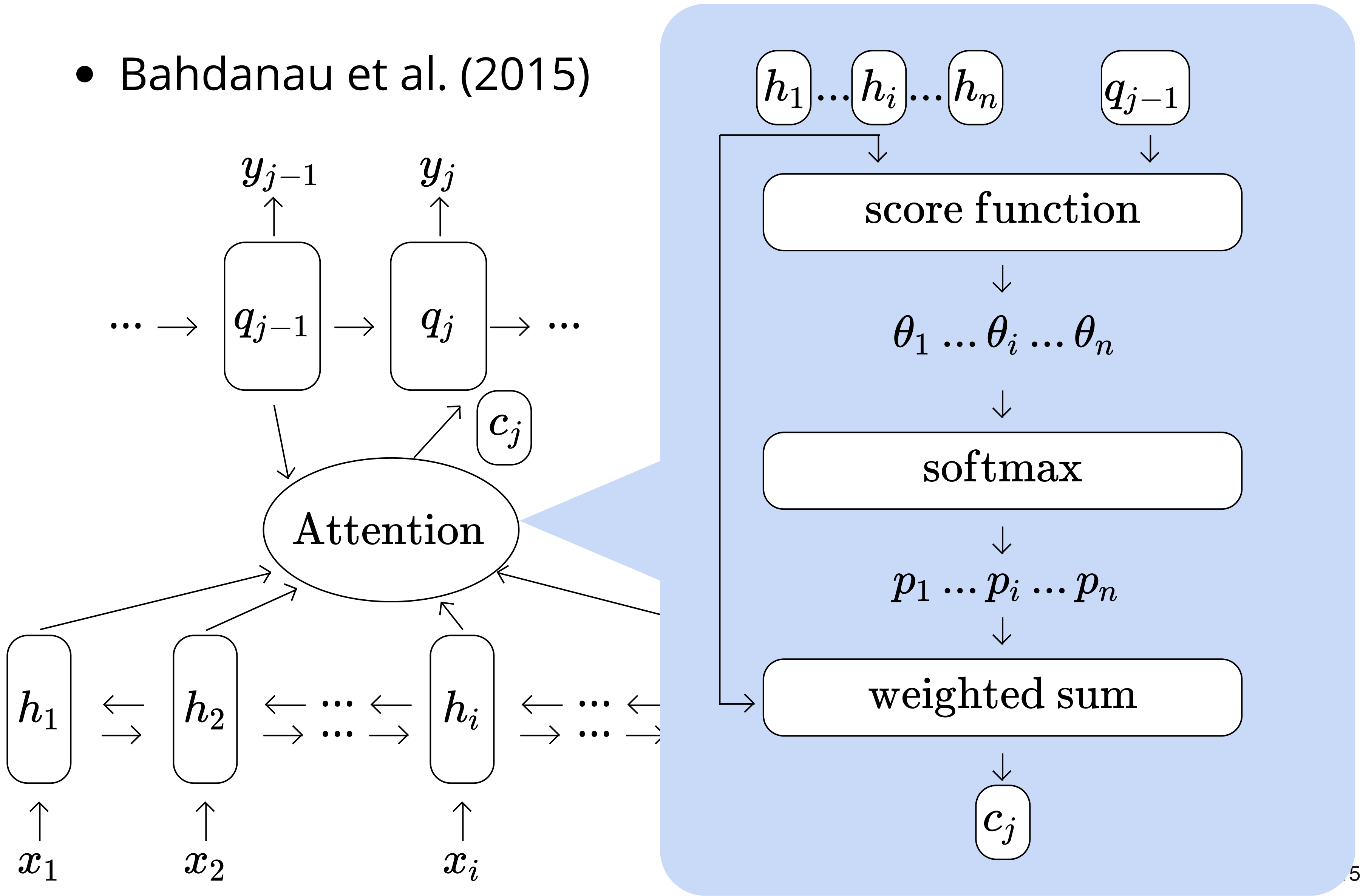
Attention mechanism

- Bahdanau et al. (2015)



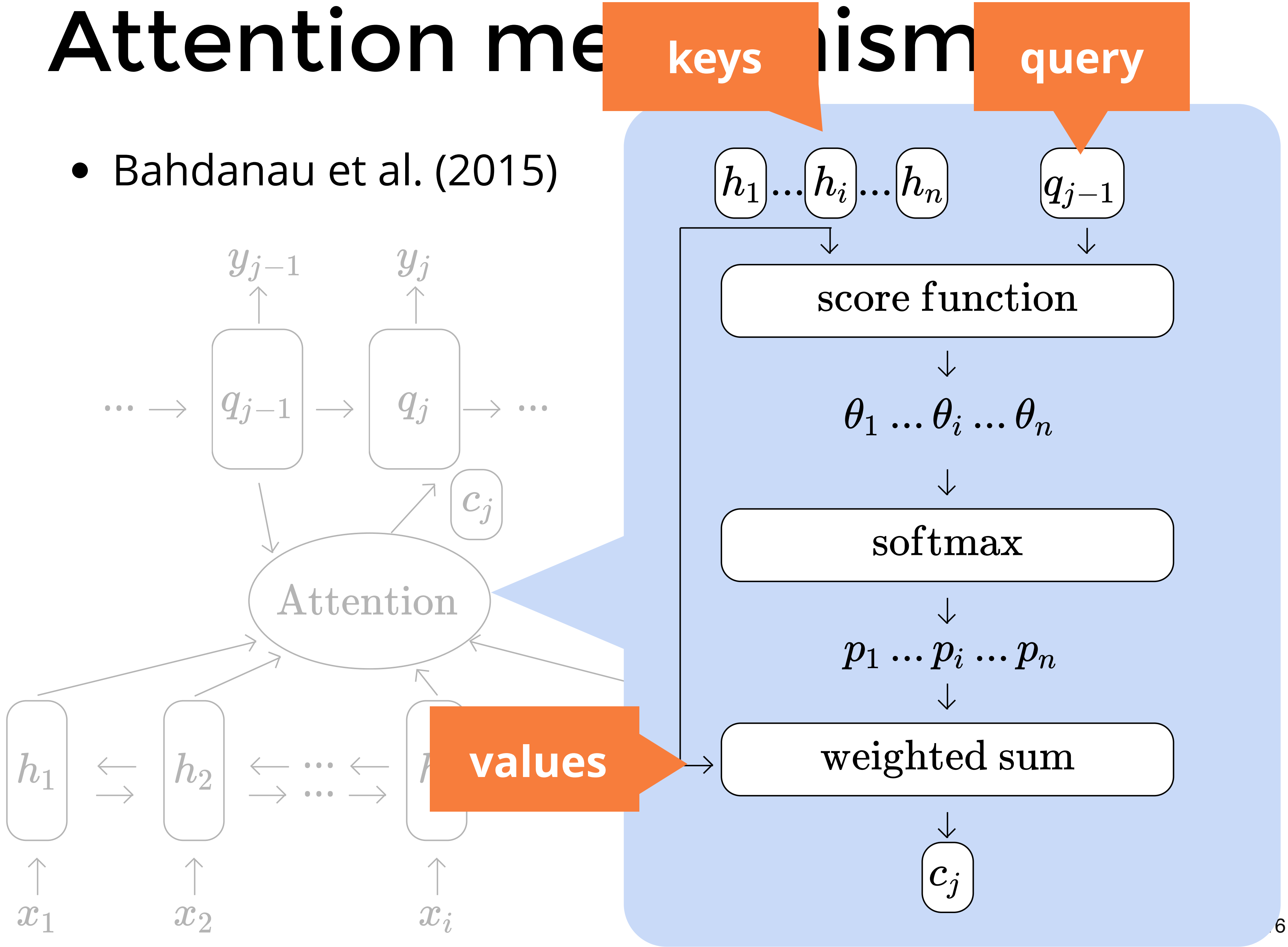
Attention mechanism

- Bahdanau et al. (2015)



Attention mechanism

- Bahdanau et al. (2015)



Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

   $\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

   $\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$

dot-product: $\mathbf{k}_j^\top \mathbf{q}, \quad (d_q == d_k)$ (Luong et al., 2015)

bilinear: $\mathbf{k}_j^\top \mathbf{W} \mathbf{q}, \quad \mathbf{W} \in \mathbb{R}^{d_k \times d_q}$ (Luong et al., 2015)

additive: $\mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{k}_j + \mathbf{W}_2 \mathbf{q})$ (Bahdanau et al., 2015)

neural net: $\mathrm{MLP}(\mathbf{q}, \mathbf{k}_j); \quad \mathrm{CNN}(\mathbf{q}, \mathbf{K}); \quad \ldots$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

   $\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$

2. Map scores to probabilities

   $\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

   $\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$

2. Map scores to probabilities

   $\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$

softmax: $\exp(\boldsymbol{\theta}_j) / \sum_k \exp(\boldsymbol{\theta}_k)$

sparsemax: $\mathrm{argmin}_{\mathbf{p} \in \triangle^n} ||\mathbf{p} - \boldsymbol{\theta}||_2^2$ (Martins and Astudillo, 2016)

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

   $$\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$$

2. Map scores to probabilities

   $$\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$$

3. Combine values

   $$\mathbf{z} = \mathbf{V}^\top \mathbf{p} = \sum_{i=1}^{m} \mathbf{V}_i \mathbf{p}_i \in \mathbb{R}^{d_v}$$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

$$\boldsymbol{\theta} = \mathrm{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$$

2. Map scores to probabilities

$$\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$$

not necessarily in the simplex! e.g.

$$\mathbf{p} = \mathrm{sigmoid}(\boldsymbol{\theta})$$

3. Combine values

$$\mathbf{z} = \mathbf{V}^\top \mathbf{p} = \sum_{i=1}^{m} \mathbf{V}_i \mathbf{p}_i \in \mathbb{R}^{d_v}$$

Attention mechanism

query

$\mathbf{q} \in \mathbb{R}^{d_q}$

keys

$\mathbf{K} \in \mathbb{R}^{n \times d_k}$

values

$\mathbf{V} \in \mathbb{R}^{n \times d_v}$

1. Compute a score between **q** and each **k**j

$$\boldsymbol{\theta} = \text{score}(\mathbf{q}, \mathbf{K}) \in \mathbb{R}^n$$

2. Map scores to probabilities

$$\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$$

but in this lecture:

$$\sum_i \mathbf{p}_i = 1$$

$$\forall i, \mathbf{p}_i \geq 0$$

3. Combine values

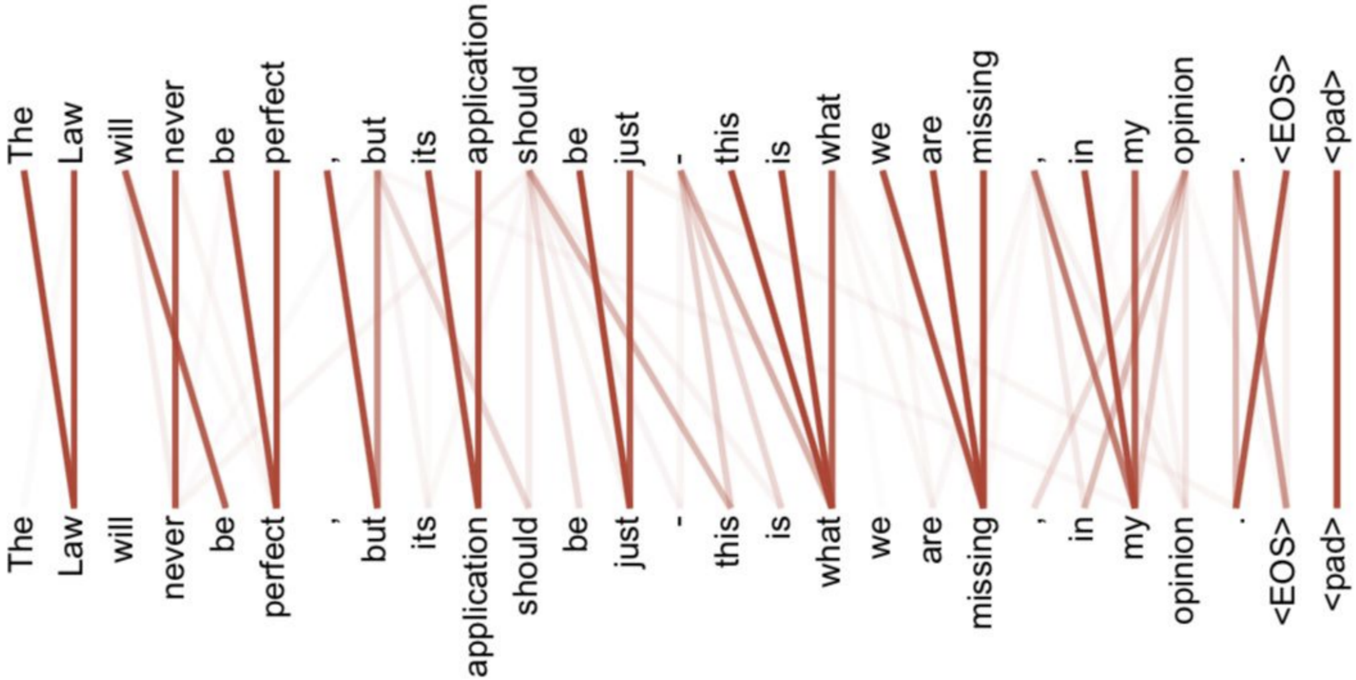
$$\mathbf{z} = \mathbf{V}^\top \mathbf{p} = \sum_{i=1}^{m} \mathbf{V}_i \mathbf{p}_i \in \mathbb{R}^{d_v}$$

Attention mechanism

```
def attention(query, keys, values=None):
    """
    query.shape is (batch_size, 1, d)
    keys.shape is (batch_size, n, d)
    values.shape is (batch_size, n, d)
    """
    # use keys as values
    if values is None:
        values = keys

    # STEP 1. scores.shape is (batch_size, 1, n)
    scores = torch.matmul(query, keys.transpose(-1, -2))

    # STEP 2. probas.shape is (batch_size, 1, n)
    probas = torch.softmax(scores, dim=-1)

    # STEP 3. c_vector.shape is (batch_size, 1, d)
    c_vector = torch.matmul(probas, values)

    return c_vector
```

Attention flavors

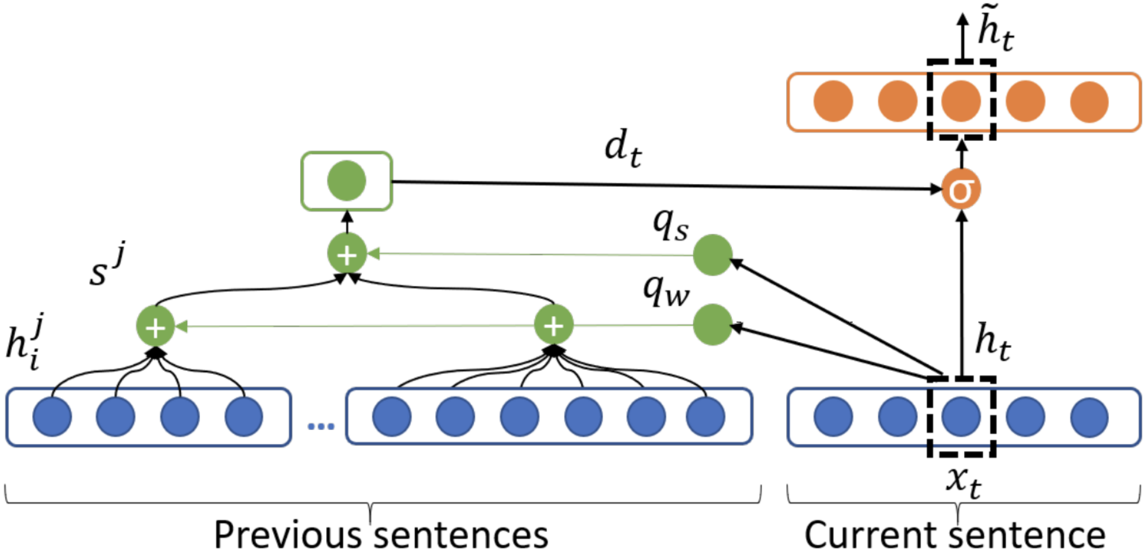
- Interaction between $\mathbf{q}, \mathbf{K}, \mathbf{V}$:
  - Self-attention: $\mathbf{q} = \mathbf{k}_j$



(Vaswani et al., 2017)

Attention flavors

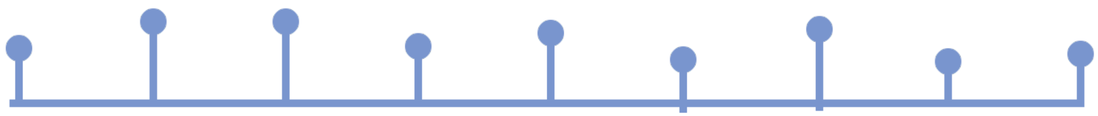
- Interaction between $\mathbf{q}, \mathbf{K}, \mathbf{V}$:
  - Hierarchical:
    - word-level $\mathbf{q}_w, \mathbf{K}_w$
    - sentence-level $\mathbf{q}_s, \mathbf{K}_s$



(Miculicich et al., 2018)

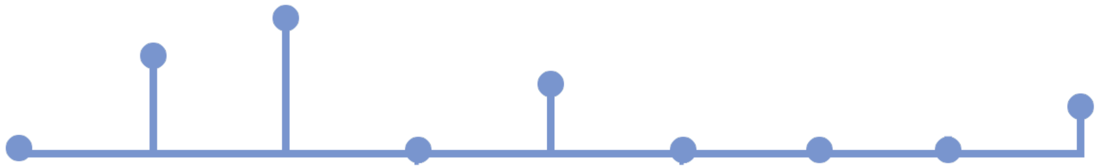
Dense vs Sparse

- $\mathbf{p} = \pi(\boldsymbol{\theta}) \in \triangle^n$



the quick brown fox jumps over the lazy dog

Dense: $|\mathrm{supp}(\mathbf{p})| = n$



the **quick brown** fox **jumps** over the lazy **dog**

Sparse: $|\mathrm{supp}(\mathbf{p})| < n$

Variational form of argmax

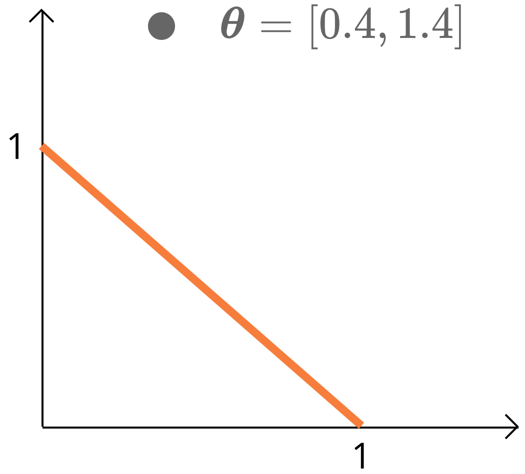
$$\max_j \boldsymbol{\theta}_j = \max_{\mathbf{p} \in \triangle^n} \mathbf{p}^\top \boldsymbol{\theta}$$

Fundamental Thm. Lin. Prog.
(Dantzig et al., 1955)

Variational form of argmax

$$\max_j \boldsymbol{\theta}_j = \max_{\mathbf{p} \in \triangle^n} \mathbf{p}^\top \boldsymbol{\theta}$$

Fundamental Thm. Lin. Prog.
(Dantzig et al., 1955)

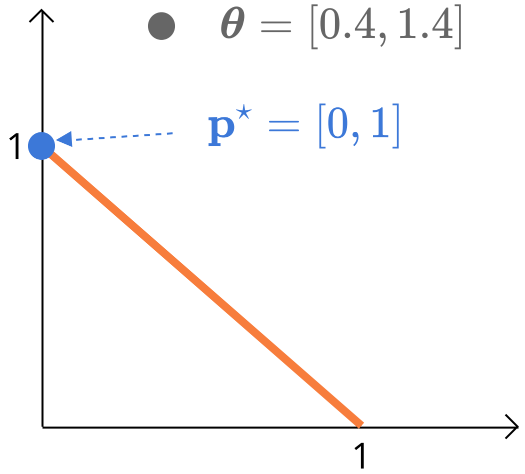


$n = 2$

Variational form of argmax

$$\max_j \boldsymbol{\theta}_j = \max_{\mathbf{p} \in \triangle^n} \mathbf{p}^\top \boldsymbol{\theta}$$

Fundamental Thm. Lin. Prog.
(Dantzig et al., 1955)

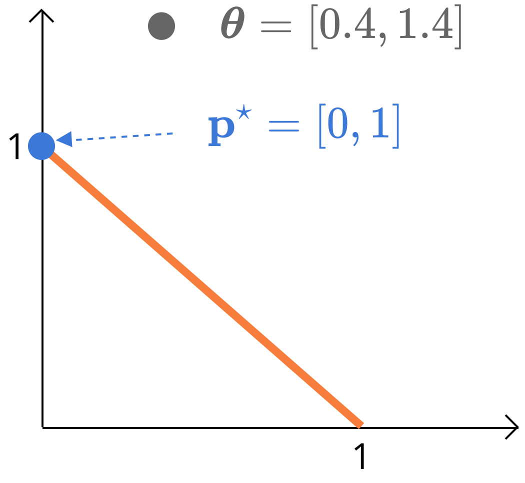


$n = 2$

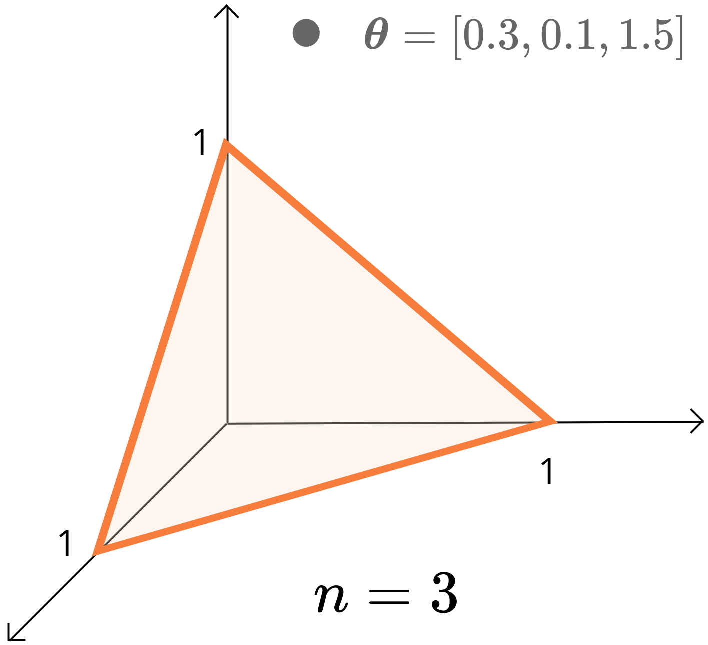
Variational form of argmax

$$\max_j \boldsymbol{\theta}_j = \max_{\mathbf{p} \in \triangle^n} \mathbf{p}^\top \boldsymbol{\theta}$$

Fundamental Thm. Lin. Prog.
(Dantzig et al., 1955)



$n = 2$

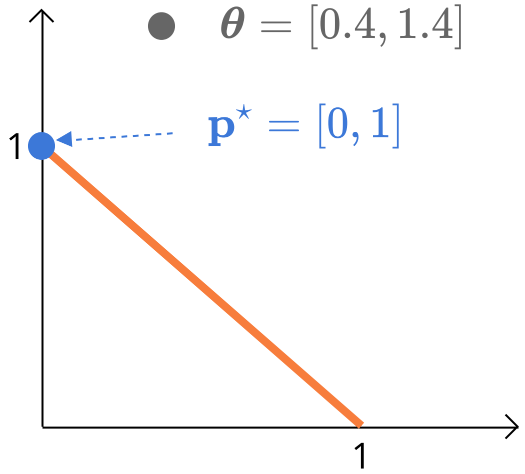


$n = 3$

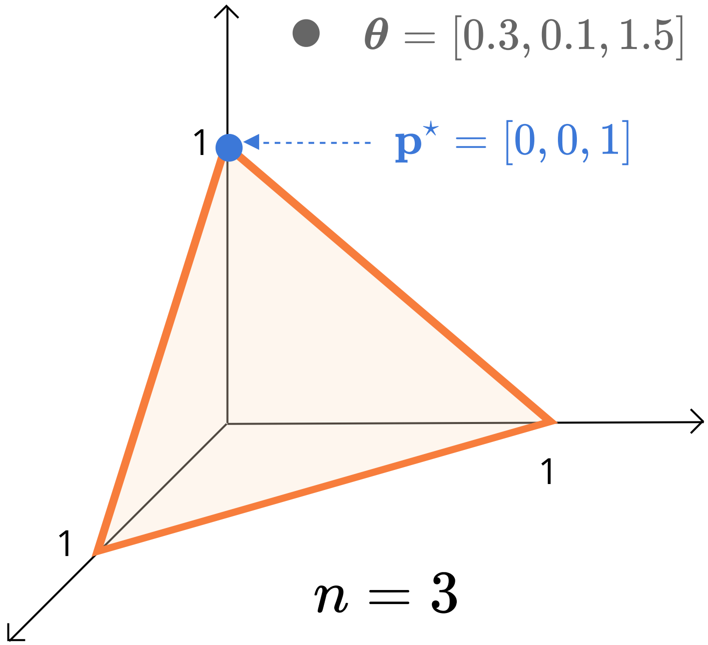
Variational form of argmax

$$\max_j \boldsymbol{\theta}_j = \max_{\mathbf{p} \in \triangle^n} \mathbf{p}^\top \boldsymbol{\theta}$$

Fundamental Thm. Lin. Prog.
(Dantzig et al., 1955)



$n = 2$

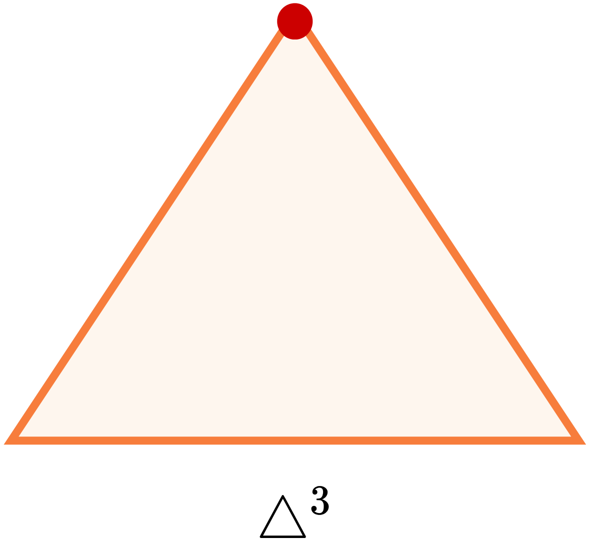


$n = 3$

Smoothed max operators

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

- argmax: $\Omega(\mathbf{p}) = 0$

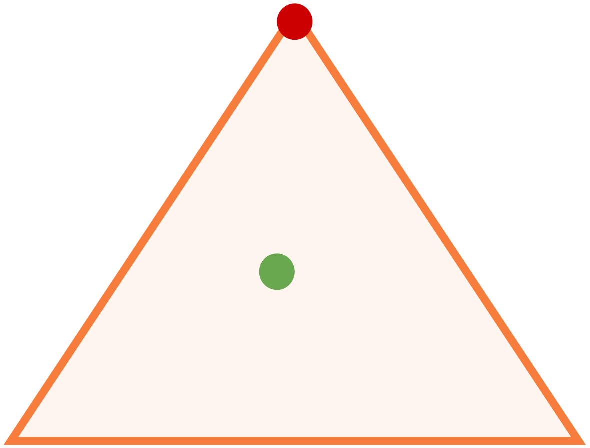


(Niculae and Blondel, 2017)

Smoothed max operators

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

- argmax: $\Omega(\mathbf{p}) = 0$
- softmax: $\Omega(\mathbf{p}) = \sum_j p_j \log p_j$

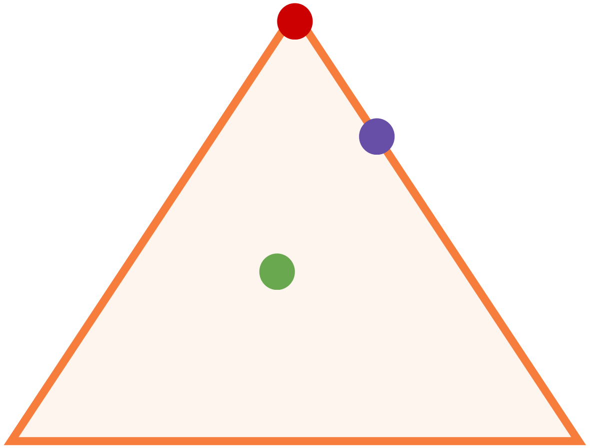


(Niculae and Blondel, 2017)

Smoothed max operators

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

- argmax: $\Omega(\mathbf{p}) = 0$
- softmax: $\Omega(\mathbf{p}) = \sum_j p_j \log p_j$
- sparsemax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2$



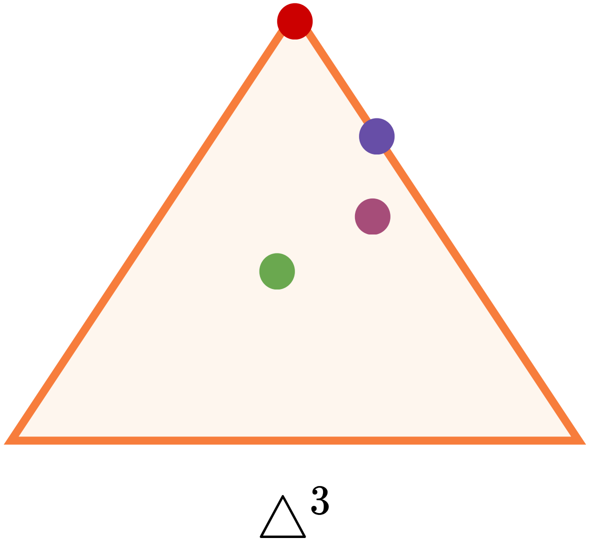
$\triangle^3$

(Niculae and Blondel, 2017)

Smoothed max operators

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

- argmax: $\Omega(\mathbf{p}) = 0$
- softmax: $\Omega(\mathbf{p}) = \sum_j p_j \log p_j$
- sparsemax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2$
- α-entmax: $\Omega(\mathbf{p}) = \frac{1}{\alpha(\alpha-1)} \sum_j p_j^\alpha$



(Niculae and Blondel, 2017)

Sparsemax and α-entmax

- sparsemax

(Martins and Astudillo, 2016)

$$\mathbf{p}^\star = [\boldsymbol{\theta} - \boldsymbol{\tau}\mathbf{1}]_+$$

Just compute $\boldsymbol{\tau}$:

$O(n \log n)$ or $O(n)$*

Sparsemax and α-entmax

- sparsemax

(Martins and Astudillo, 2016)

$$\mathbf{p}^\star = [\boldsymbol{\theta} - \boldsymbol{\tau}\mathbf{1}]_+$$

Just compute $\boldsymbol{\tau}$:

$O(n \log n)$ or $O(n)$*

- α-entmax

* (Peters, Niculae, and Martins, 2019)

$$\mathbf{p}^\star = [(\alpha - 1)\boldsymbol{\theta} - \boldsymbol{\tau}\mathbf{1}]_+^{1/(\alpha-1)}$$

Just compute $\boldsymbol{\tau}$:

$O(n \log n)$ or $O(n)$*

Sparsemax and α-entmax

- sparsemax

(Martins and Astudillo, 2016)

$$\mathbf{p}^\star = [\boldsymbol{\theta} - \boldsymbol{\tau}\mathbf{1}]_+$$

Just compute $\boldsymbol{\tau}$:

$O(n \log n)$ or $O(n)$*

- α-entmax

* (Peters, Niculae, and Martins, 2019)

$$\mathbf{p}^\star = [(\alpha - 1)\boldsymbol{\theta} - \boldsymbol{\tau}\mathbf{1}]_+^{1/(\alpha-1)}$$

Just compute $\boldsymbol{\tau}$:

$O(n \log n)$ or $O(n)$*

Jacobian:

$$\mathbf{J}_{\alpha-\mathrm{entmax}} = \mathrm{diag}(\mathbf{s}) - \frac{1}{||\mathbf{s}||_1}\mathbf{s}\mathbf{s}^\top$$

$$s_j = \begin{cases} (p_j^\star)^{2-\alpha}, & \text{if } p_j^\star > 0 \\ 0, & \text{otherwise} \end{cases}$$

Sparsemax and α-entmax

$$p^\star = [.99, .01, 0] \implies \mathbf{s} = [1, 1, 0]$$
$$p^\star = [.50, .50, 0] \implies \mathbf{s} = [1, 1, 0]$$

The Jacobian of sparsemax ($\alpha = 2$) depends only on the support and not on the actual values of $\mathbf{p}^\star$

Jacobian:

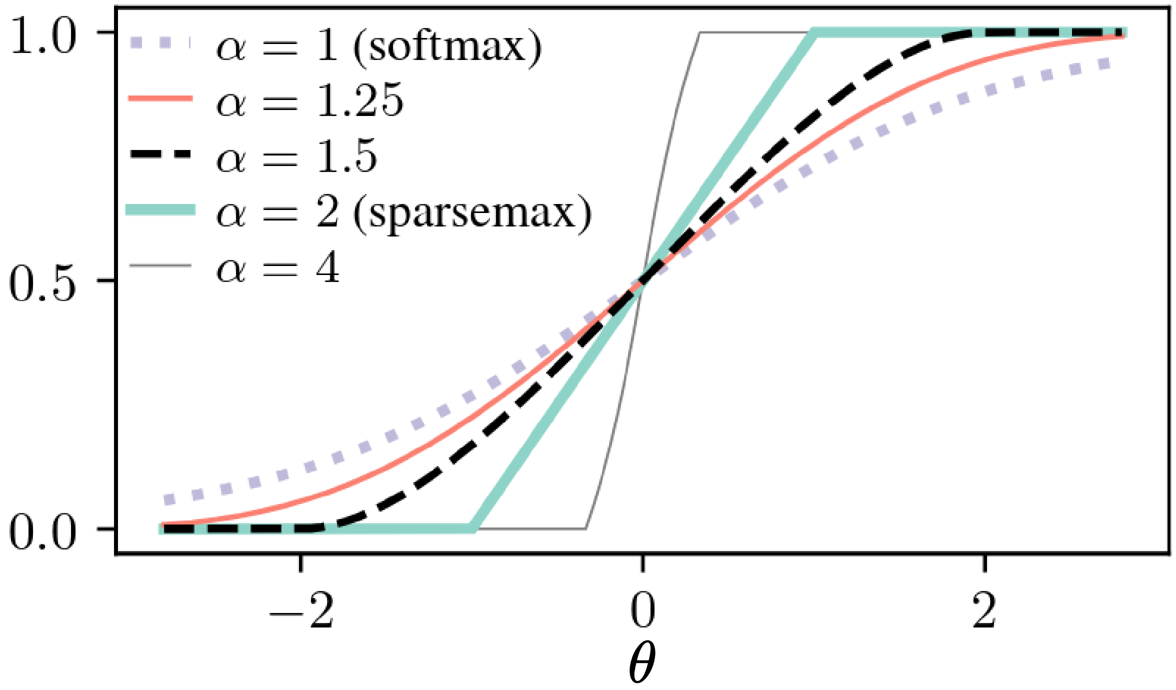
$$\mathbf{J}_{\alpha\mathrm{-entmax}} = \mathrm{diag}(\mathbf{s}) - \frac{1}{||\mathbf{s}||_1}\mathbf{s}\mathbf{s}^\top$$

$$s_j = \begin{cases} (p_j^\star)^{2-\alpha}, & \text{if } p_j^\star > 0 \\ 0, & \text{otherwise} \end{cases}$$

Sparsemax and α-entmax

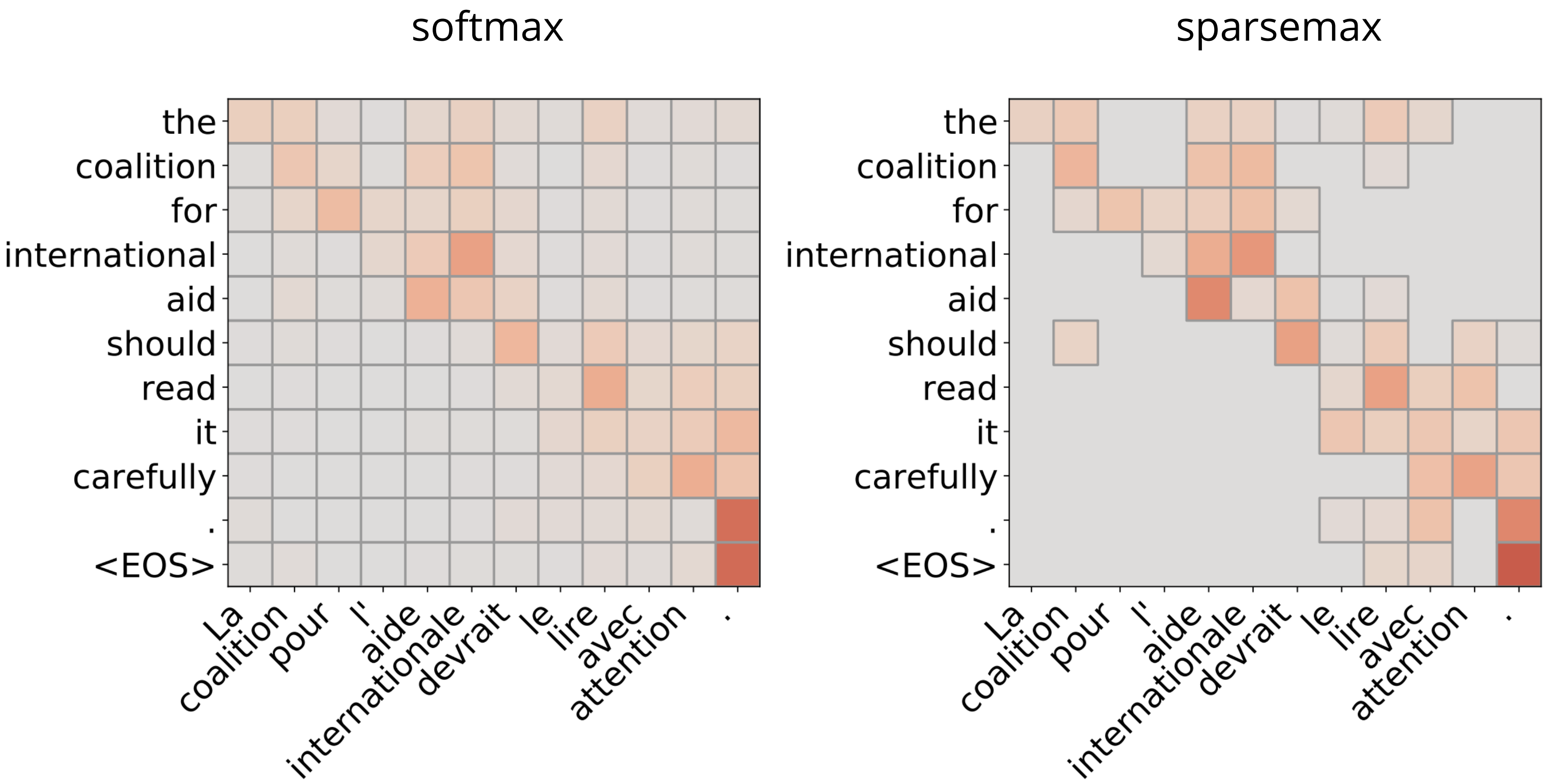
$$\alpha\text{-entmax}(\boldsymbol{\theta}) := \underset{\mathbf{p}\in\triangle^n}{\arg\max}\, \mathbf{p}^\top\boldsymbol{\theta} + H_\alpha(\mathbf{p})$$

Tsallis α-entropy regularizer (Tsallis, 1988)



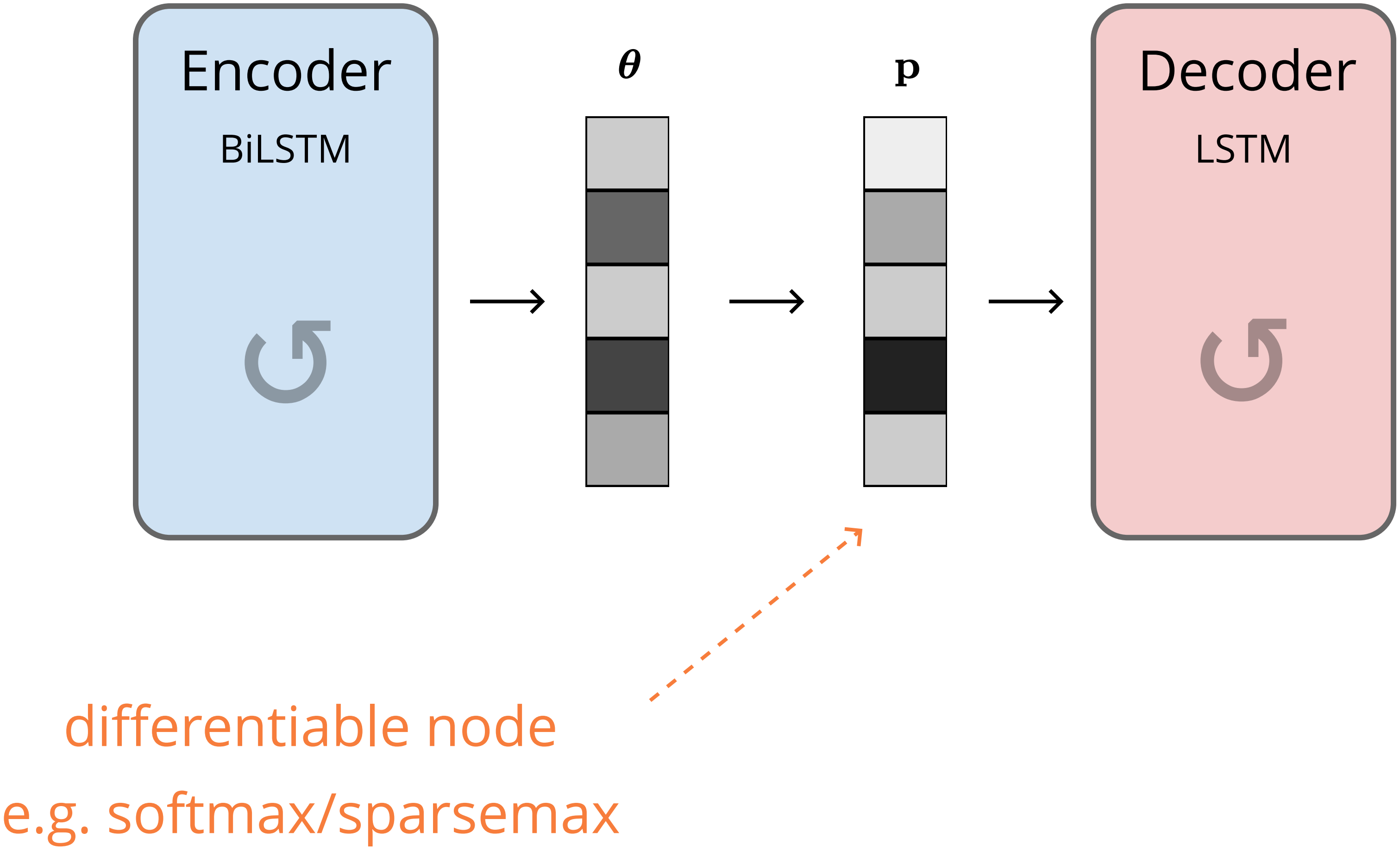
(Peters, Niculae, and Martins, 2019)

Sparsemax and α-entmax

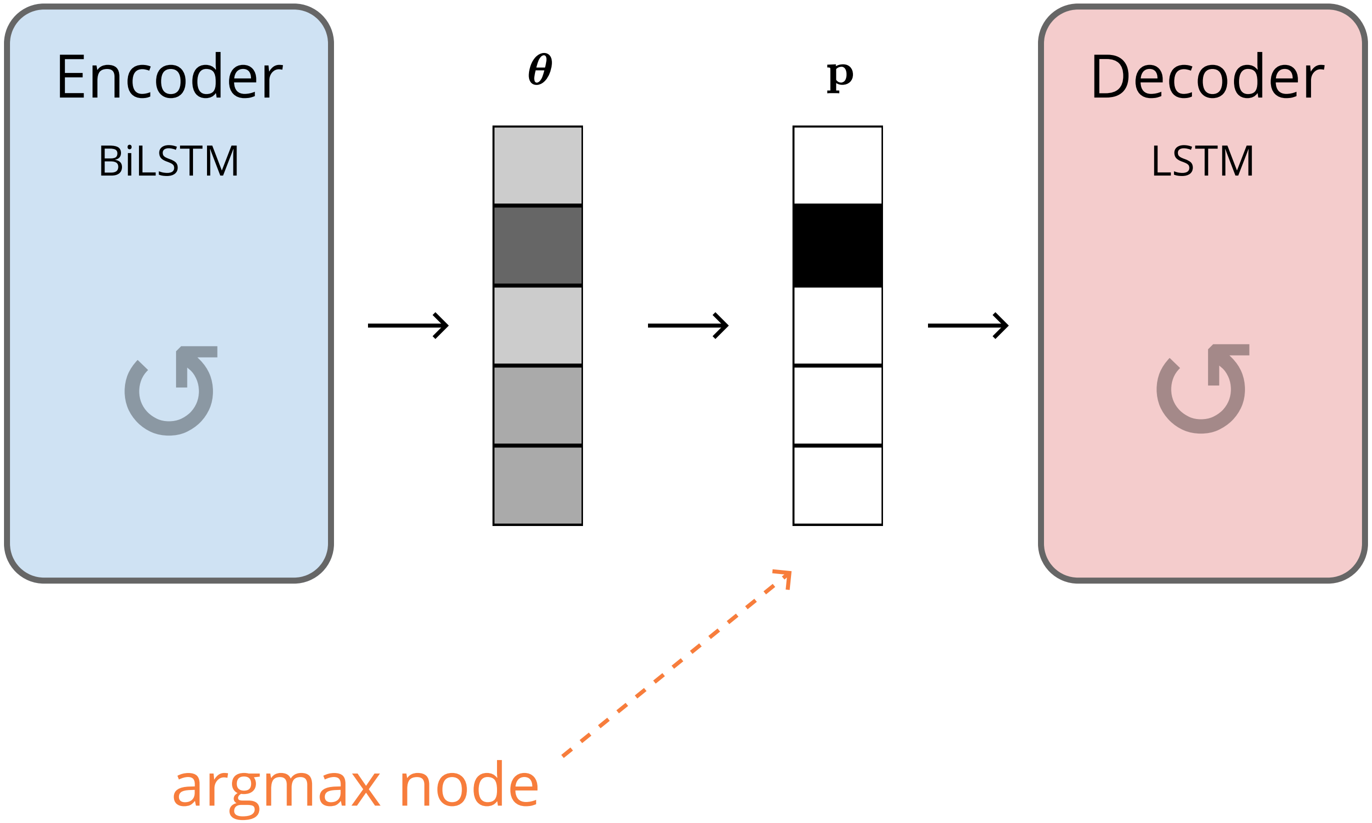


(Niculae and Blondel, 2017)

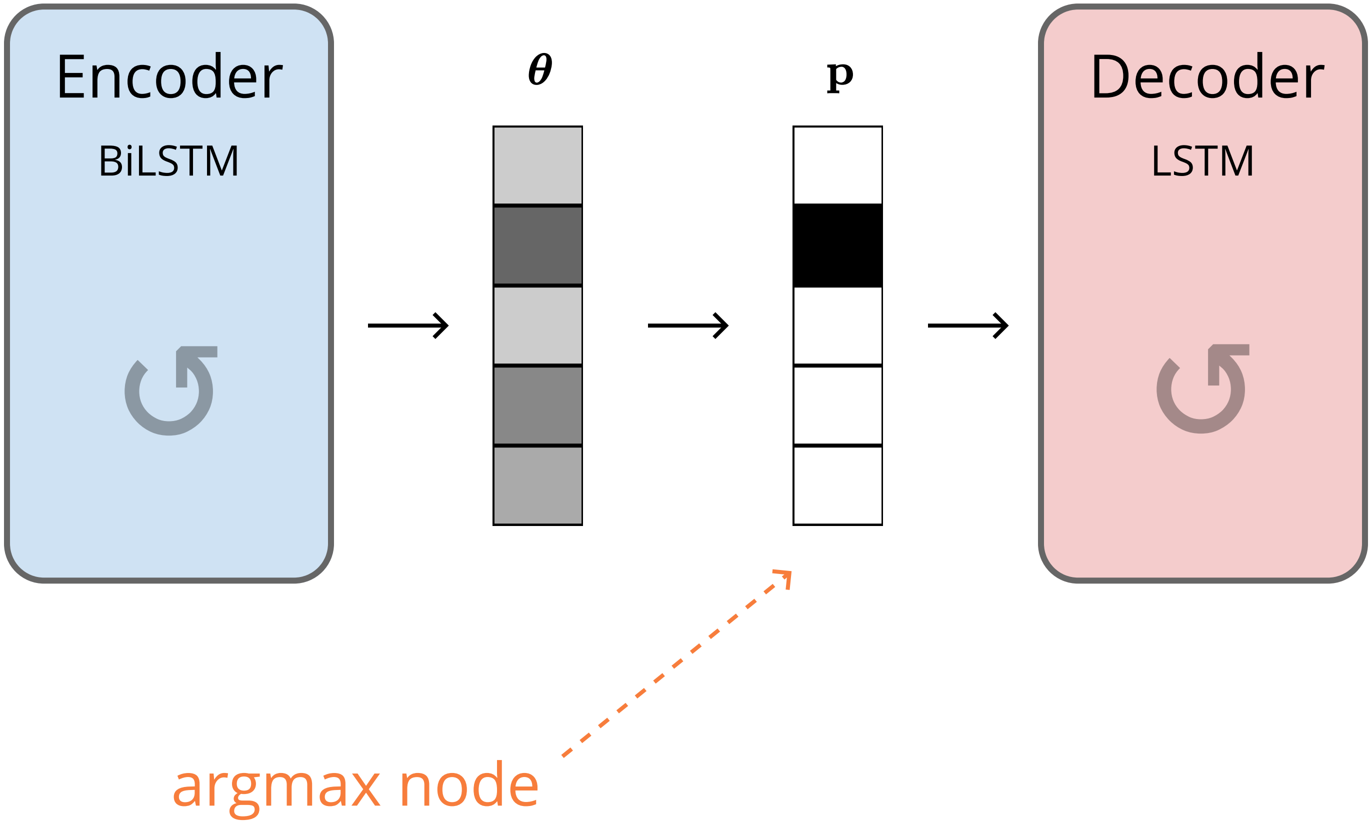
Soft attention



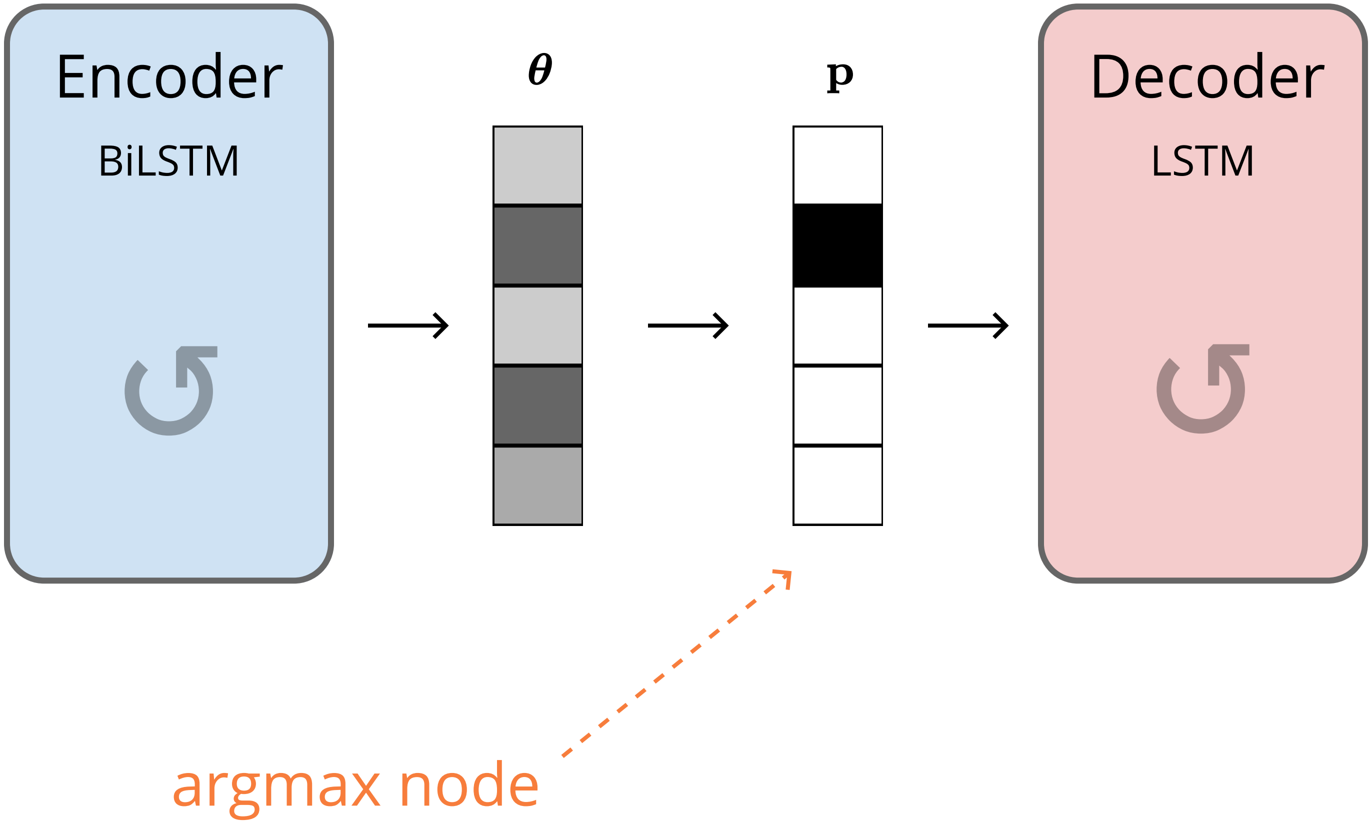
Hard attention



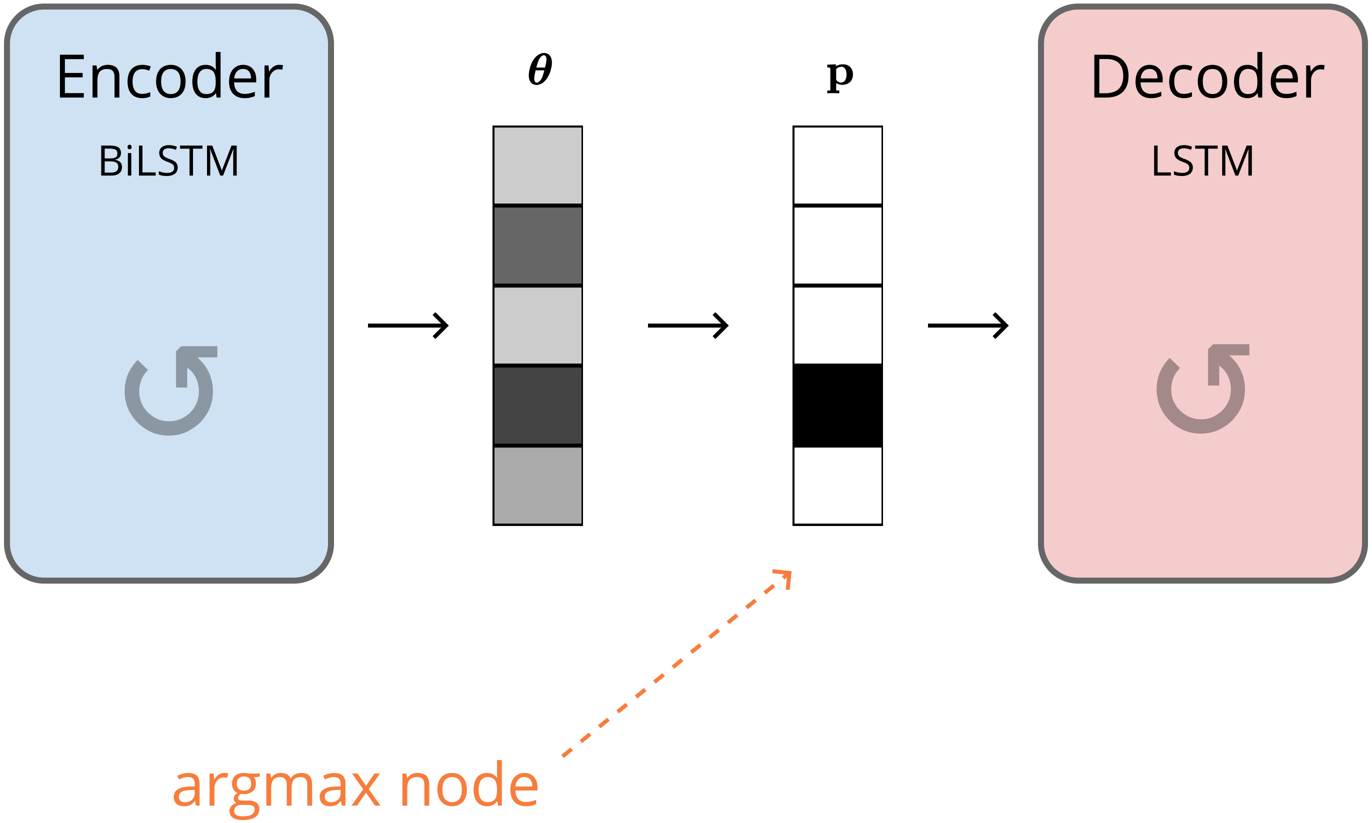
Hard attention



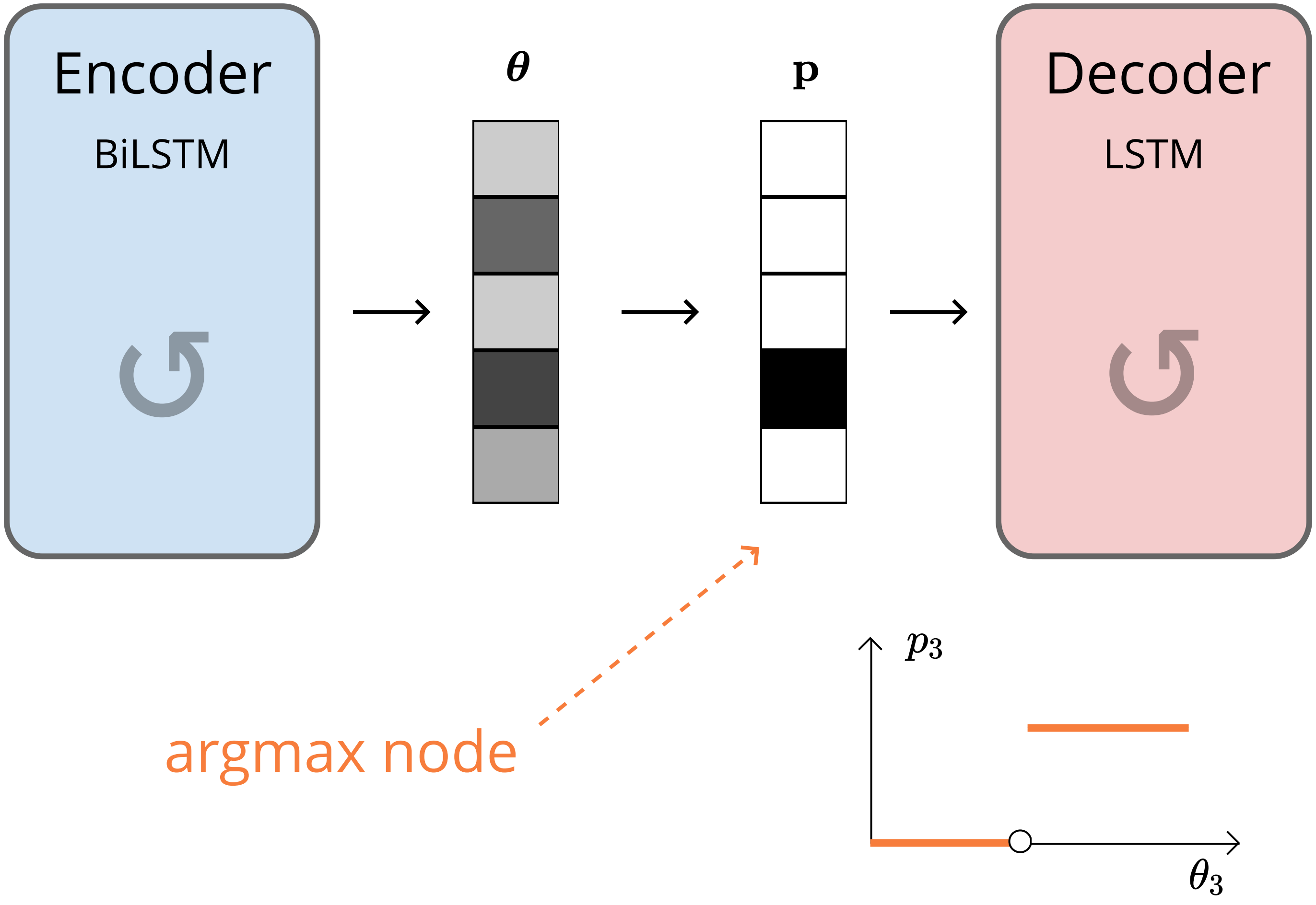
Hard attention



Hard attention



Hard attention



Soft vs Hard

Soft	**Hard**
"smooth selection"	*"subset selection"*
Continuous representation	Discrete representation
"soft" decisions	"binary" decisions
Differentiable	Non differentiable
Just backprop!	REINFORCE / surrogate gradients / reparameterization trick / perturb-and-MAP / etc.

Xu et al. (2015)
Niculae et al. (2018)
Mihaylova et al. (2020)

Structured attention

- Structural biases?

  I am going to the store → Vou à loja

- When you generate “Vou”, where do you attend?

  I am going to the store → **Vou** à loja

  I am going **to the** store → Vou **à** loja

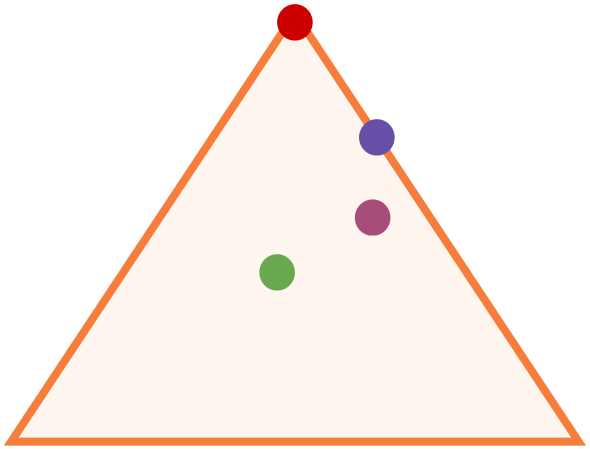
  I am going to the **store** → Vou à **loja**

- Can we consider the sequential structure of our input/output?
  - Note: $\boldsymbol{\pi}(\boldsymbol{\theta}) \in \triangle^n$

Fusedmax

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

- argmax: $\Omega(\mathbf{p}) = 0$
- softmax: $\Omega(\mathbf{p}) = \sum_j p_j \log p_j$
- sparsemax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2$
- α-entmax: $\Omega(\mathbf{p}) = \frac{1}{\alpha(\alpha-1)} \sum_j p_j^\alpha$
- fusedmax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2 + \sum_j |p_j - p_{j-1}|$



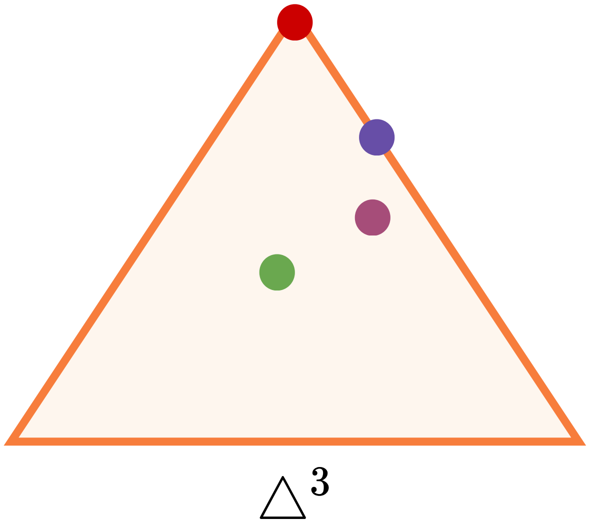
$\triangle^3$

(Niculae and Blondel, 2017)

Fusedmax

$$\boldsymbol{\pi}_\Omega(\boldsymbol{\theta}) = \underset{\mathbf{p} \in \triangle^n}{\arg\max}\, \mathbf{p}^\top \boldsymbol{\theta} - \Omega(\mathbf{p})$$

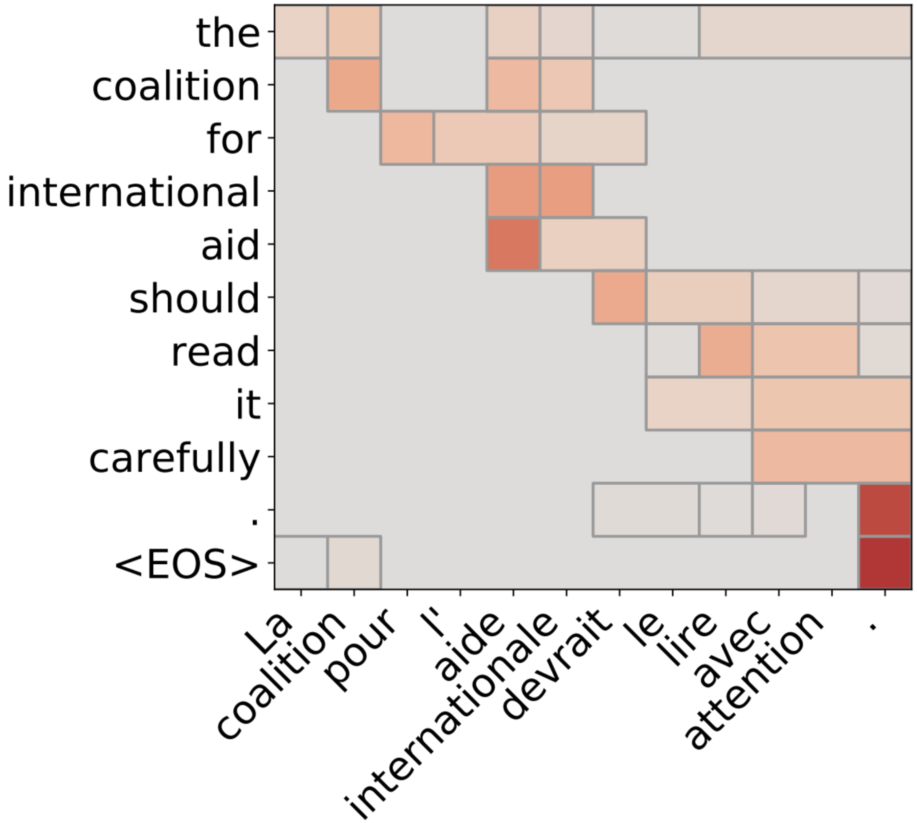
- argmax: $\Omega(\mathbf{p}) = 0$
- softmax: $\Omega(\mathbf{p}) = \sum_j p_j \log p_j$
- sparsemax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2$
- α-entmax: $\Omega(\mathbf{p}) = \frac{1}{\alpha(\alpha-1)} \sum_j p_j^\alpha$
- fusedmax: $\Omega(\mathbf{p}) = \frac{1}{2}||\mathbf{p}||_2^2 + \sum_j |p_j - p_{j-1}|$



penalize weight differences between adjacent positions

(Niculae and Blondel, 2017)

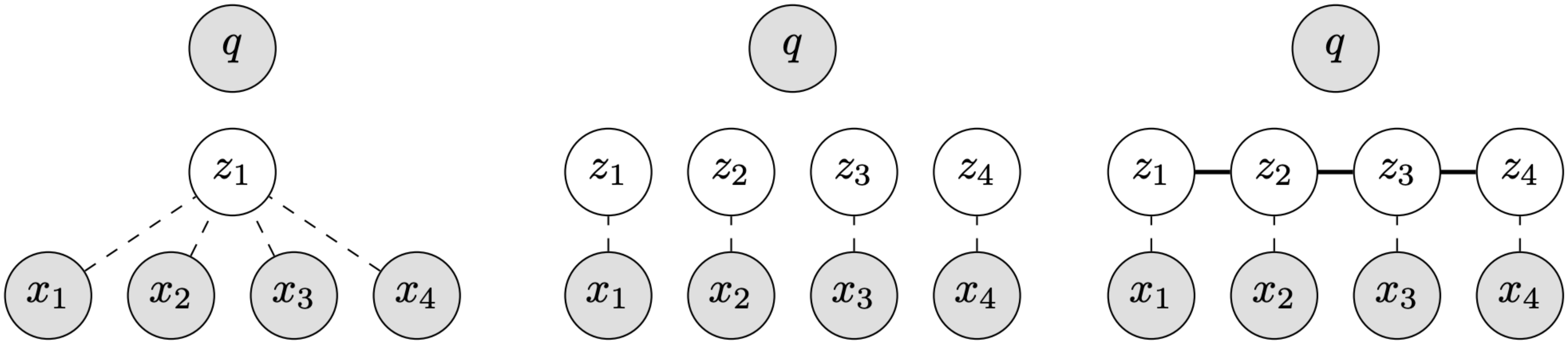
Fusedmax



(Niculae and Blondel, 2017)

Latent structured attention

- Consider binary variables (sigmoids) z_i instead of $\boldsymbol{\pi}(\boldsymbol{\theta})$
- Structured: linear dependencies on z_i
  - Linear-chain CRF
  - Use marginals from forward-backward



$\pi(\theta)$ | independent z_i | linear dependencies on z_i

(Kim et al., 2017)

Latent structured attention

- Consider binary variables (sigmoids) instead of $\boldsymbol{\pi}(\boldsymbol{\theta})$
- Structured: linear d
  - Lin
  - U

Cons:
no sparsity
specialized backprop implementation

$\pi(\theta)$

linear dependencies on z_i

(Kim et al., 2017)

Latent structured attention

- Consider binary variables (sigmoids) instead of $\boldsymbol{\pi}(\boldsymbol{\theta})$
- Structured: linear d
  - Lin
  - U

Cons:
no sparsity
specialized backprop implementation

An alternative: SparseMAP
structured counterpart of sparsemax
model any kind of structure (just need MAP)
plug & play implemenation

(Niculae et al., 2018)

z_4

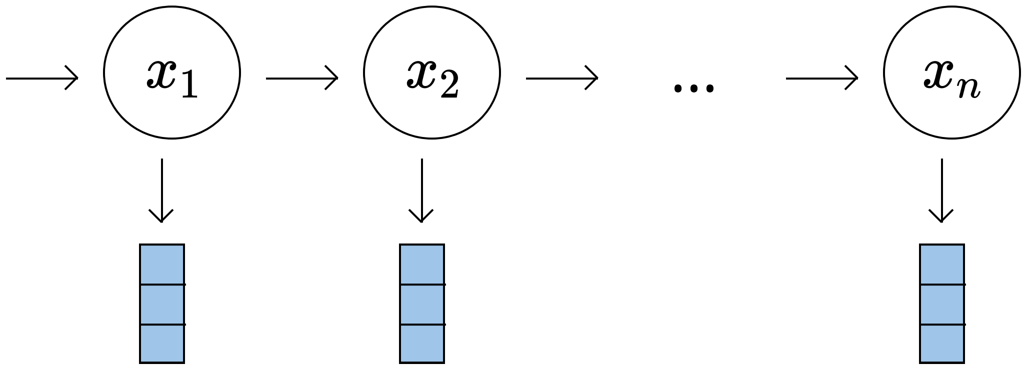
x_1 x_2 x_3 x_3 x_4

$\pi(\theta)$ ind linear dependencies on z_i

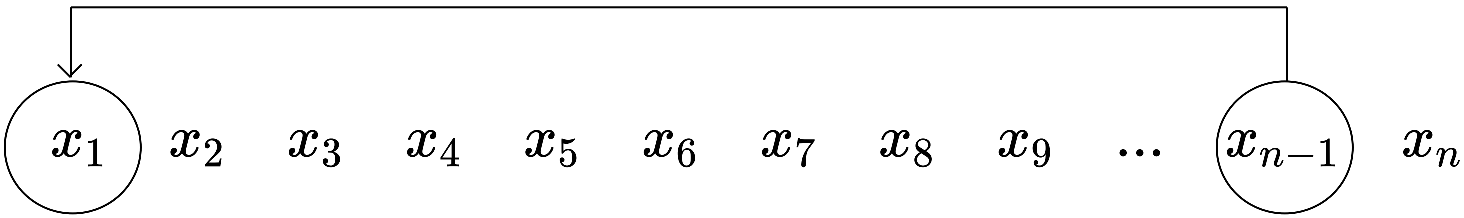
(Kim et al., 2017)

Drawbacks of RNNs

- Sequential mechanism prohibits parallelization

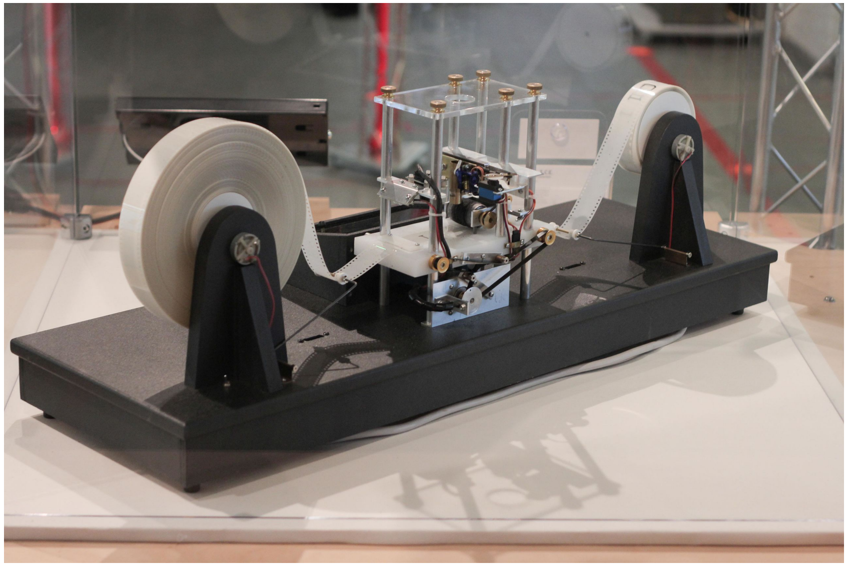


- Long-range dependencies are tricky, despite gating

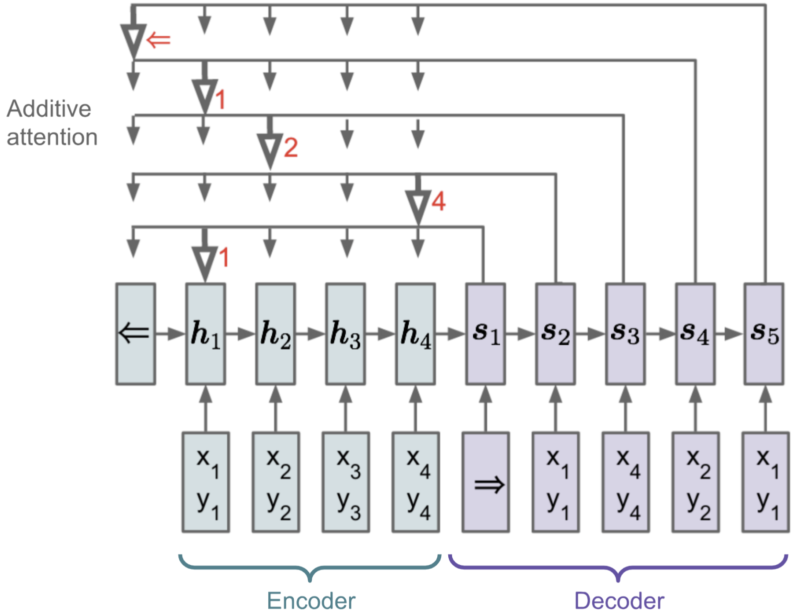


Beyond RNN-based seq2seq

- Neural Turing Machines (Graves et al. 2014)
- Memory networks (Weston et al. 2015)
- Pointer networks (Vinyals et al. 2015)
- Transformer (Vaswani et al. 2017)



(finite-tape) Turing Machine



Pointer network

Pause

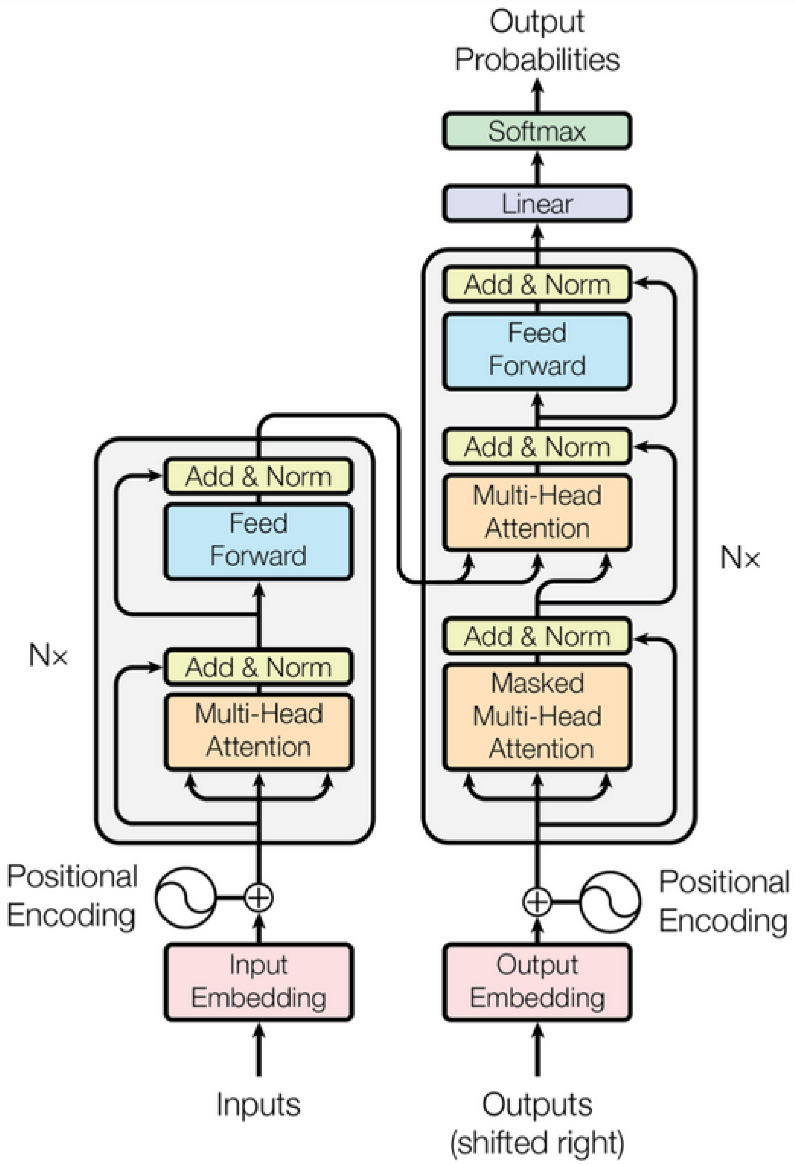
☕ 🥪 🍃

Self-attention networks

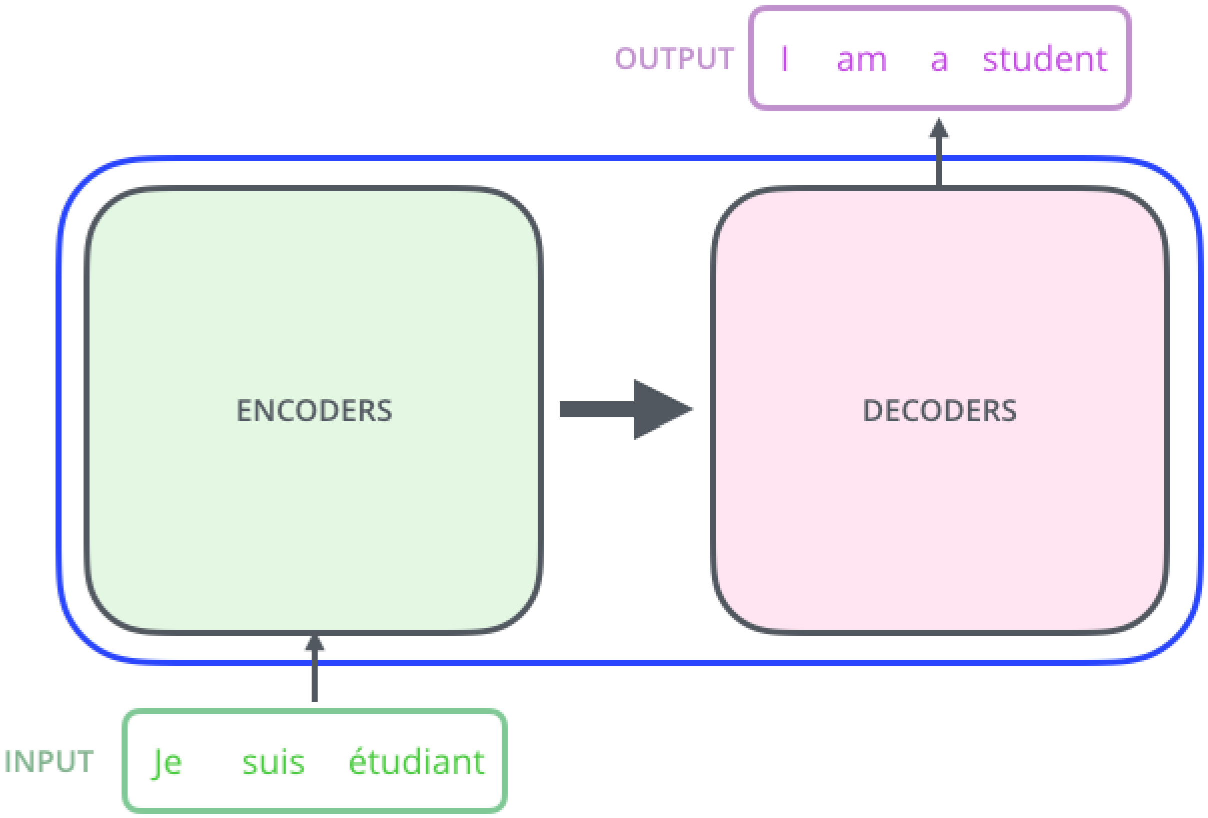
Transformer

(Vaswani et al. 2017)

Transformer

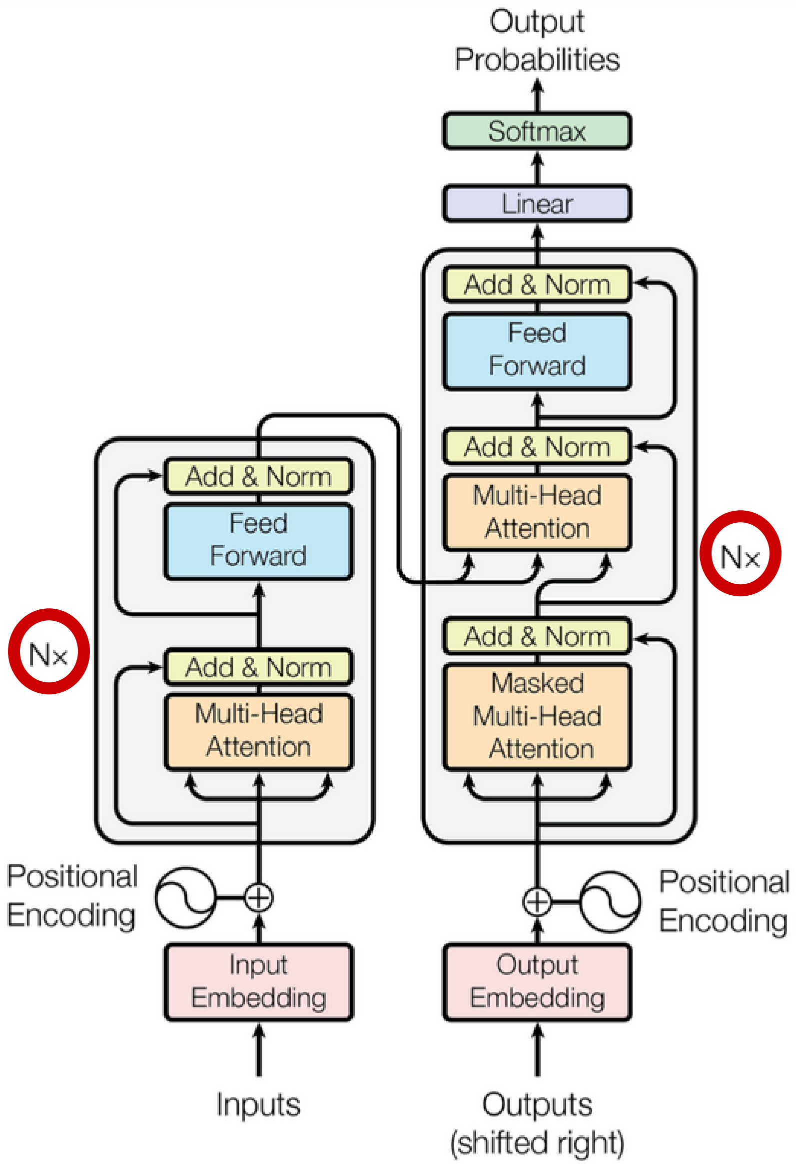


Transformer

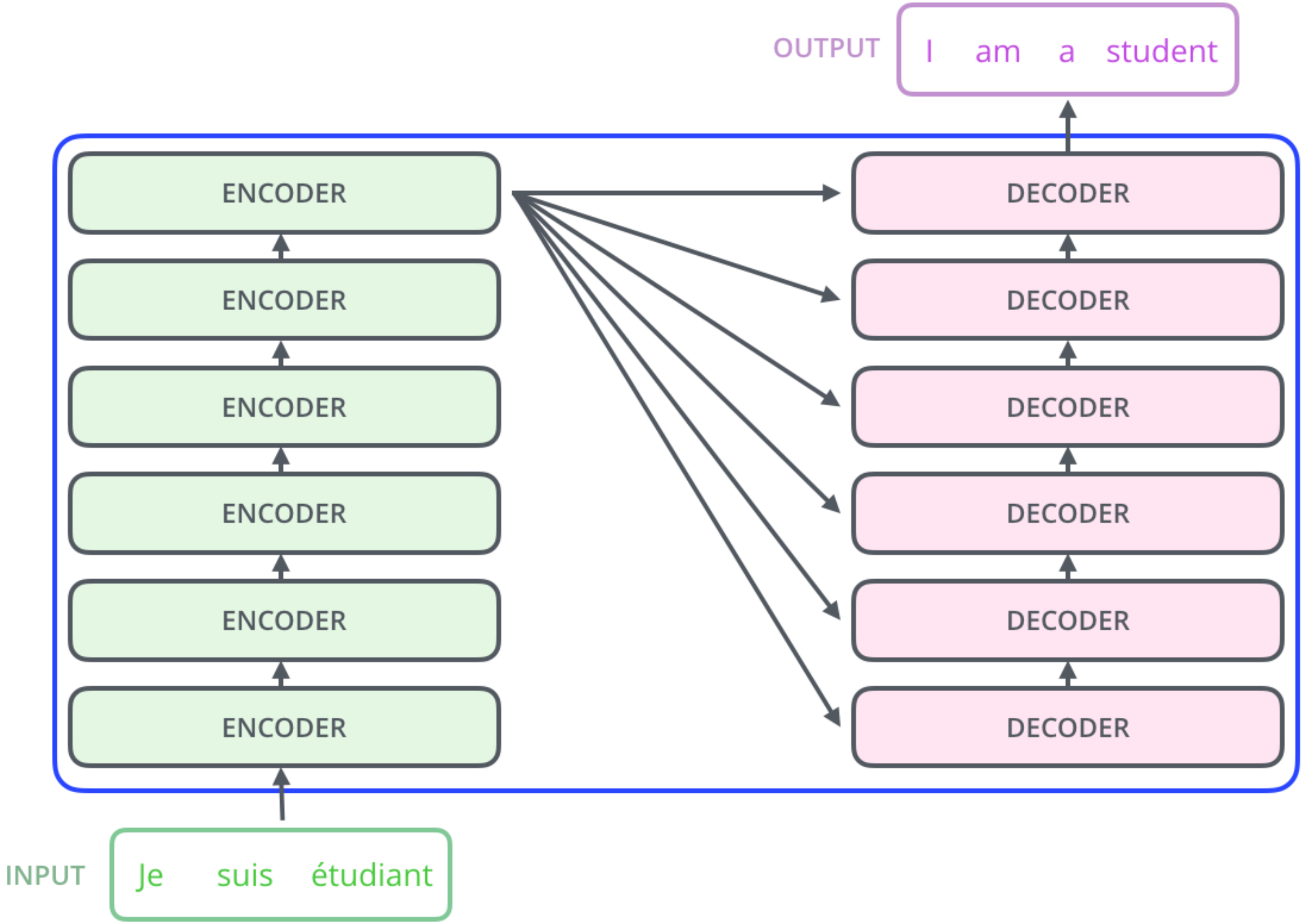


$$\boxed{x_1\, x_2 \ldots x_n} \xrightarrow{\text{encode}} \boxed{\mathbf{r}_1\, \mathbf{r}_2 \ldots \mathbf{r}_n} \xrightarrow{\text{decode}} \boxed{y_1\, y_2 \ldots y_m}$$

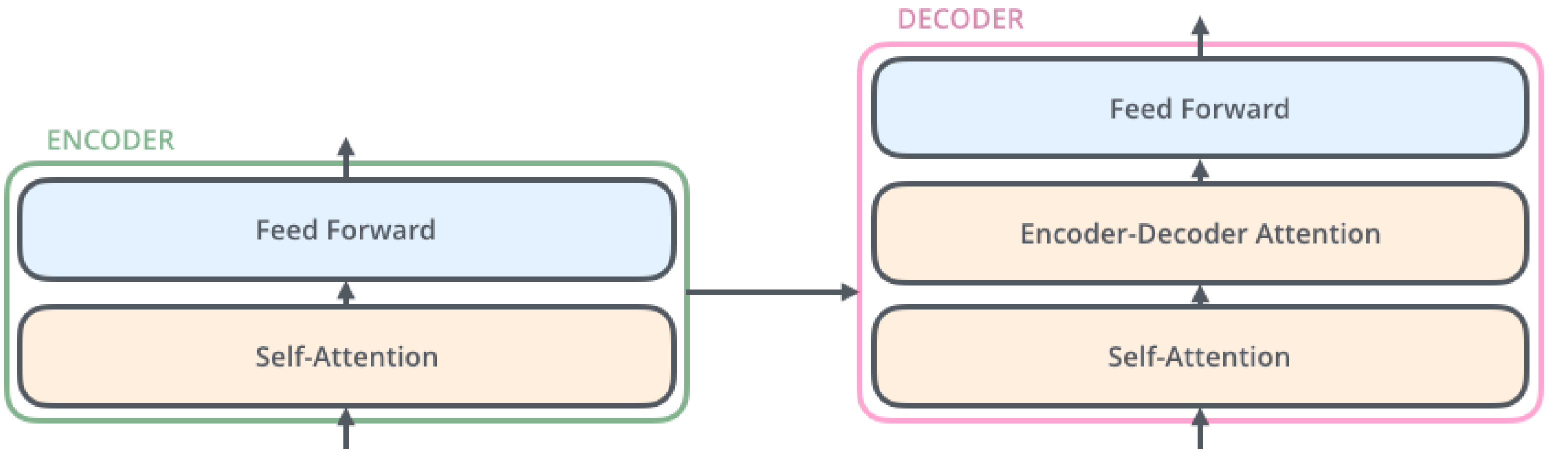
Transformer



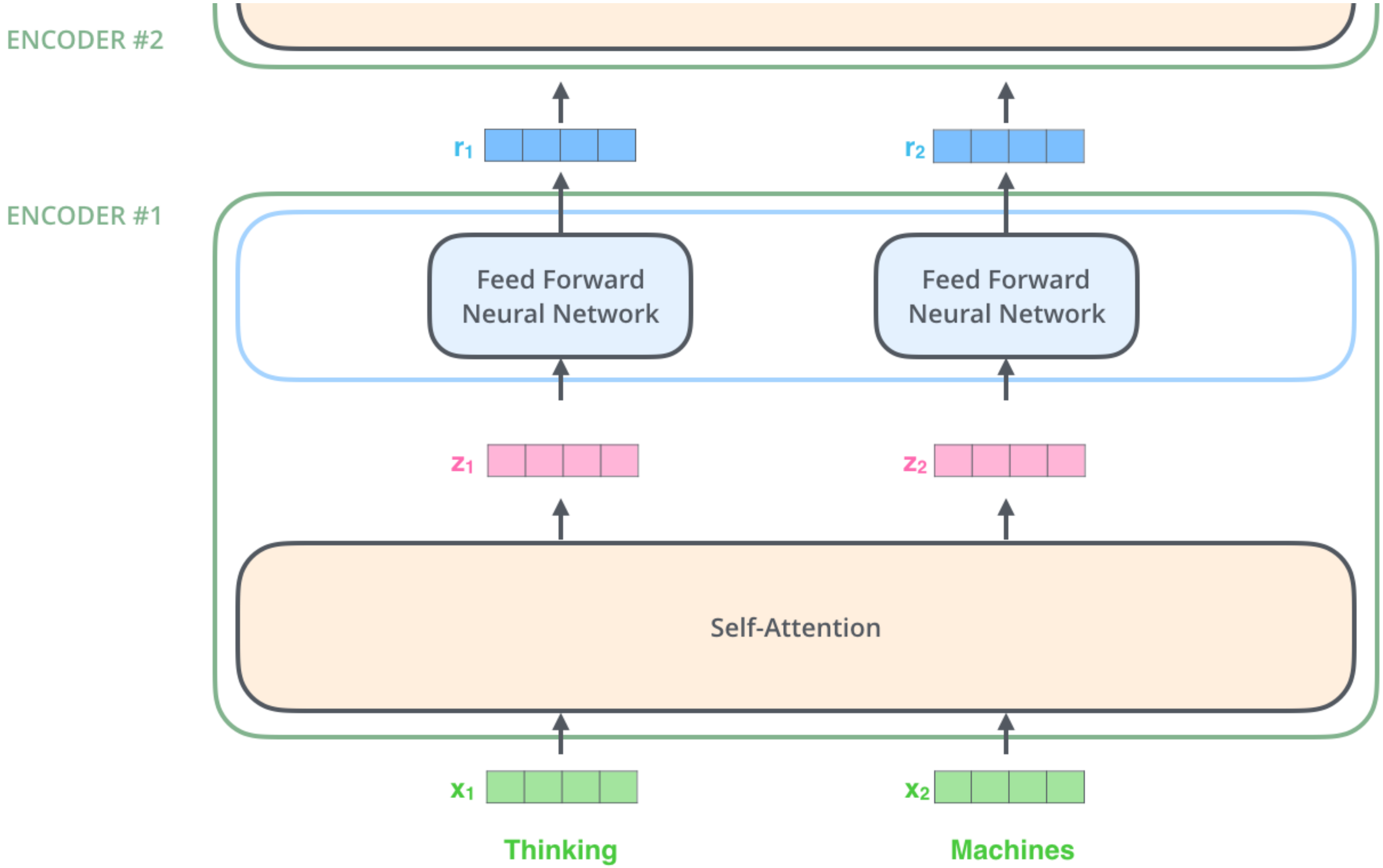
Transformer



Transformer blocks

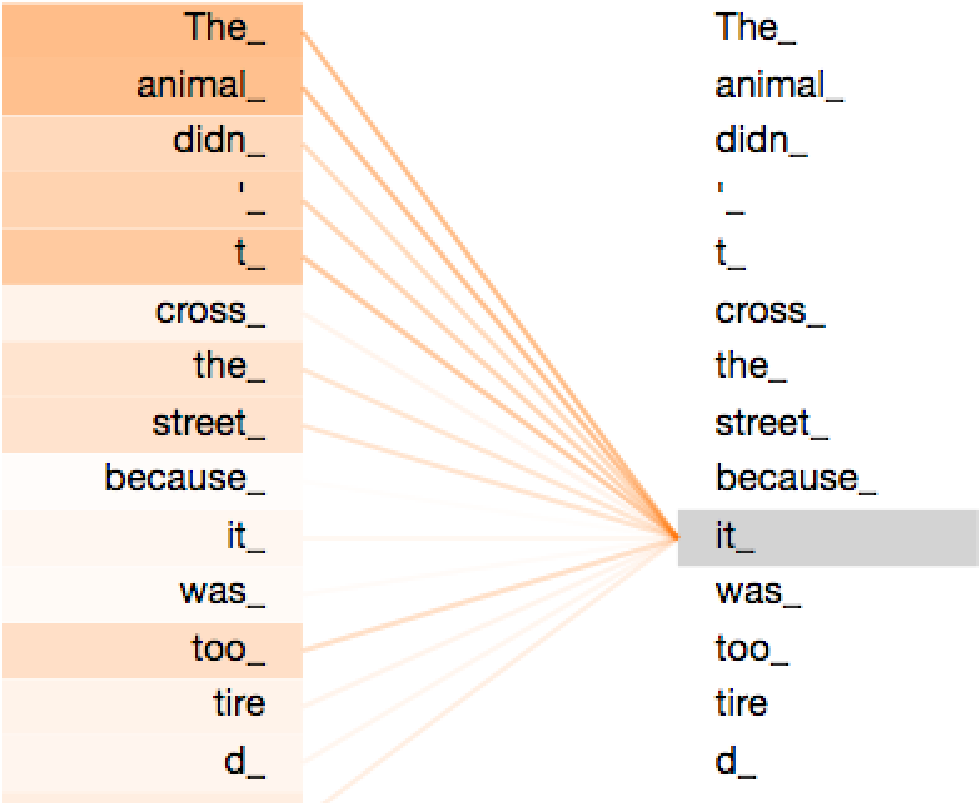


The encoder


ENCODER #2
r1
r2
ENCODER #1
Feed Forward
Neural Network
Feed Forward
Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

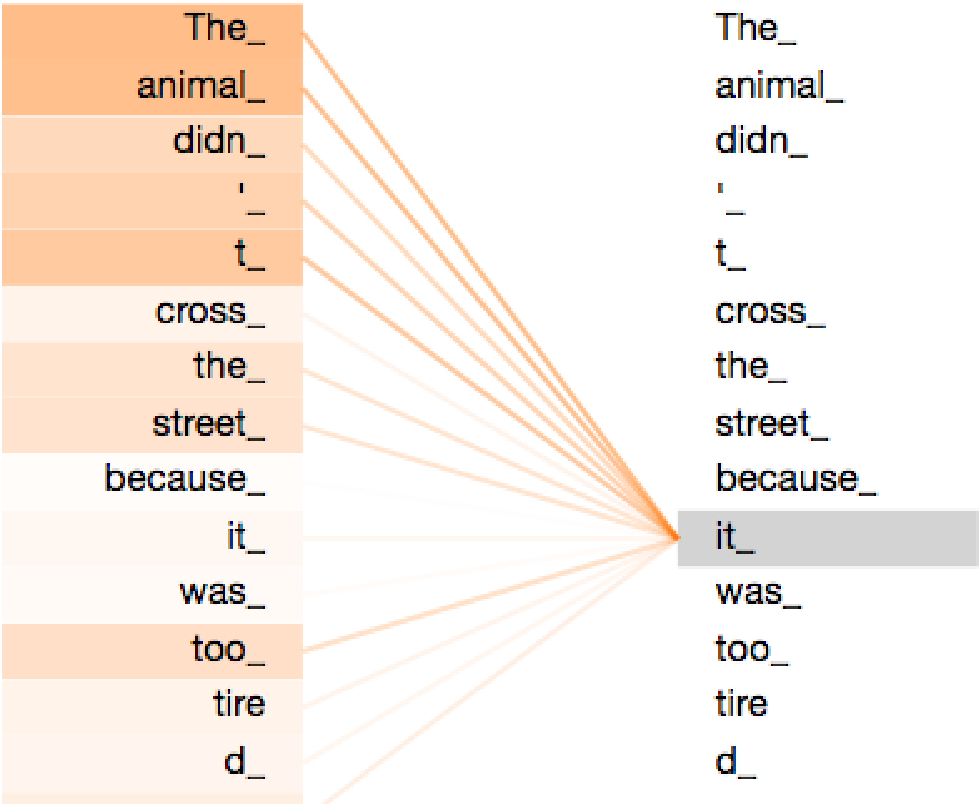
Self-attention

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



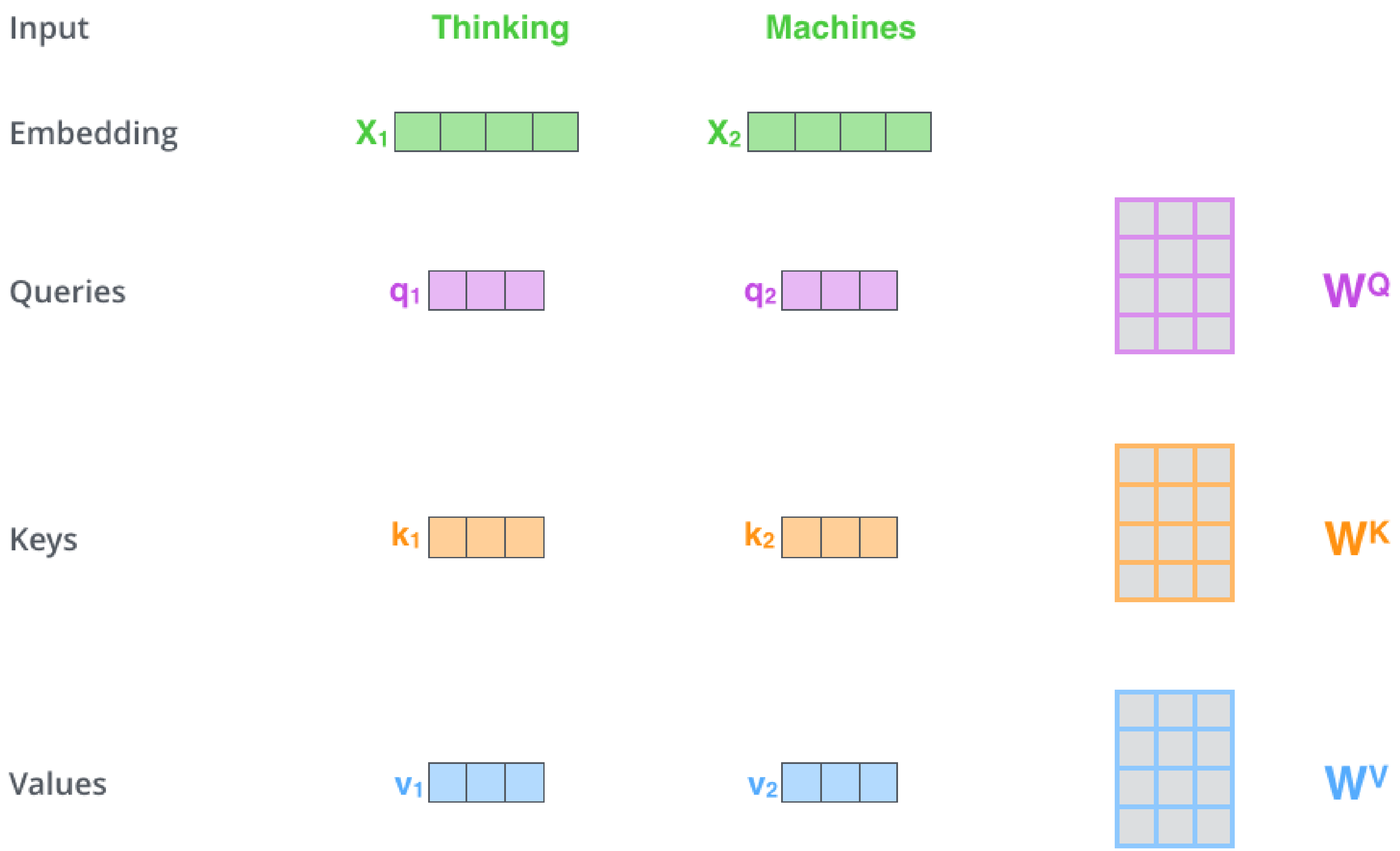
Self-attention

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

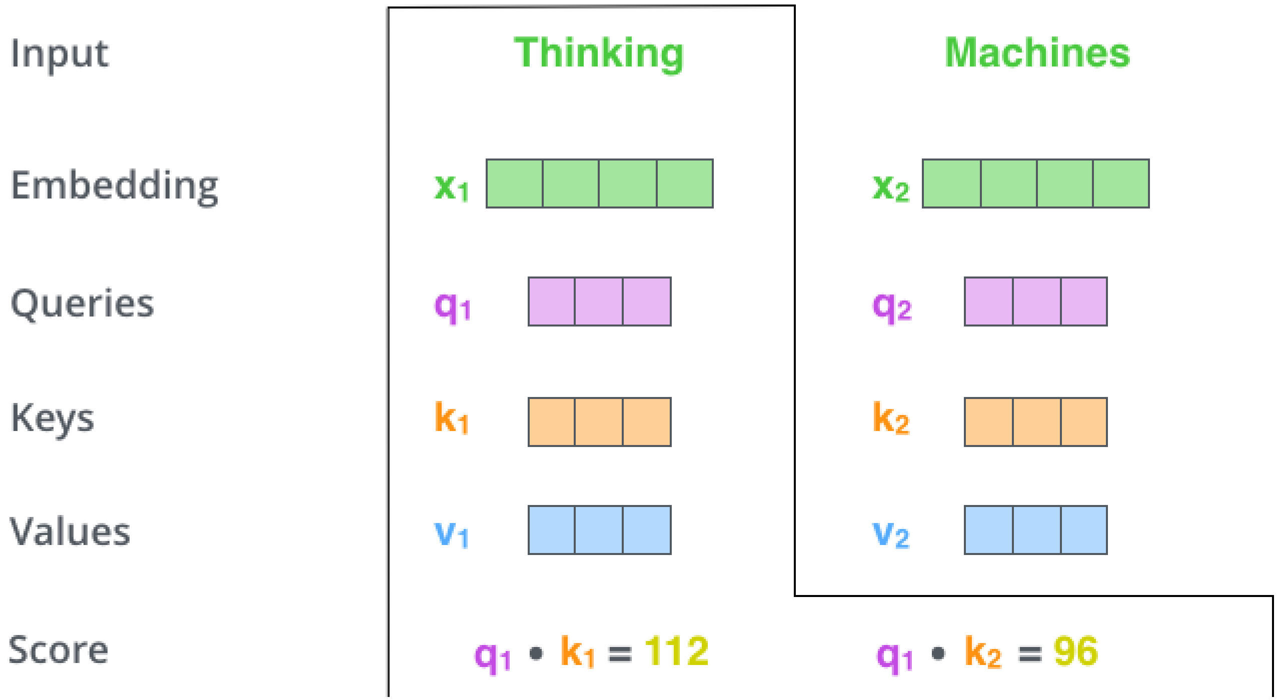


$\mathbf{Q}_j = \mathbf{K}_j = \mathbf{V}_j \in \mathbb{R}^d \quad \Longleftrightarrow \quad$ dot-product scorer!

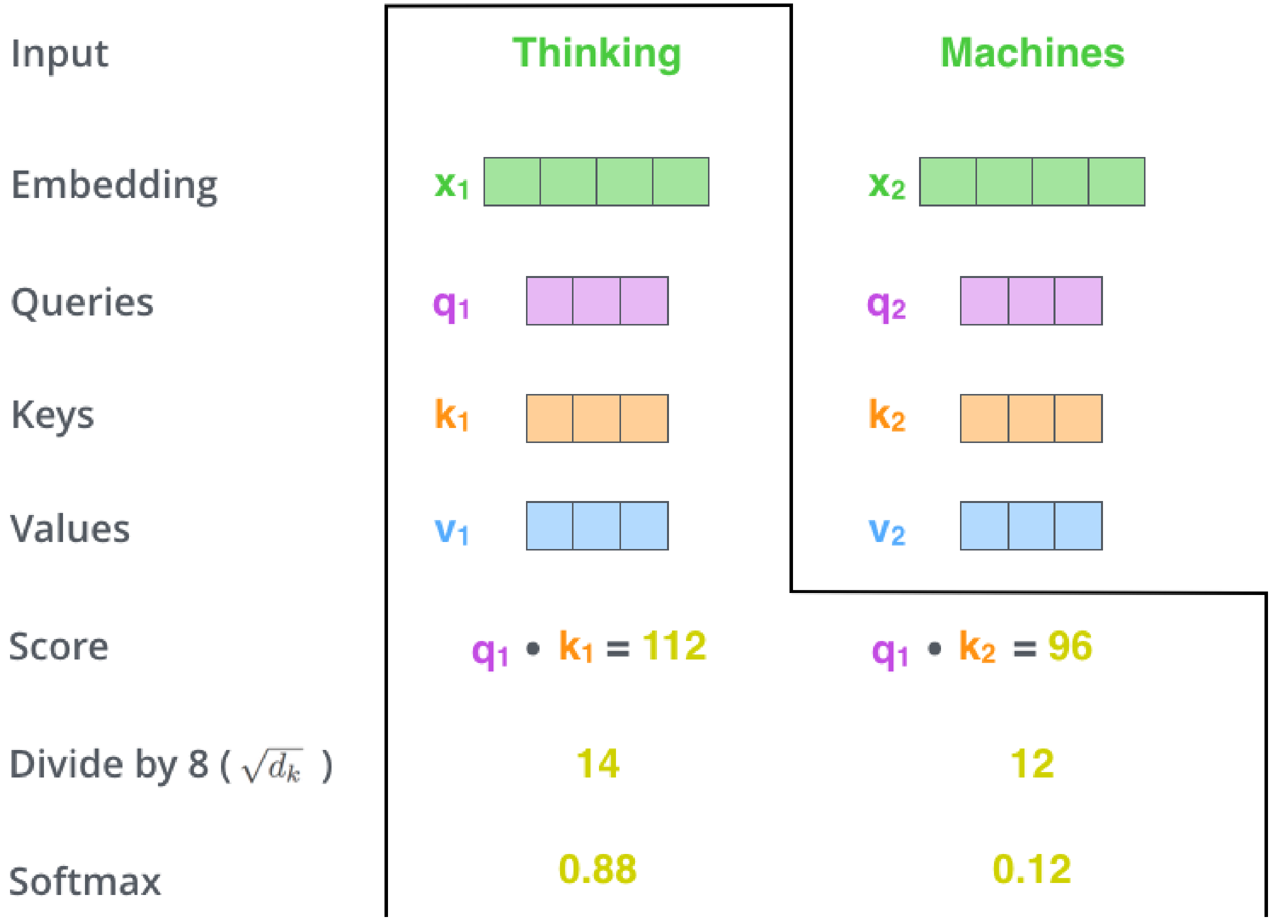
Transformer self-attention

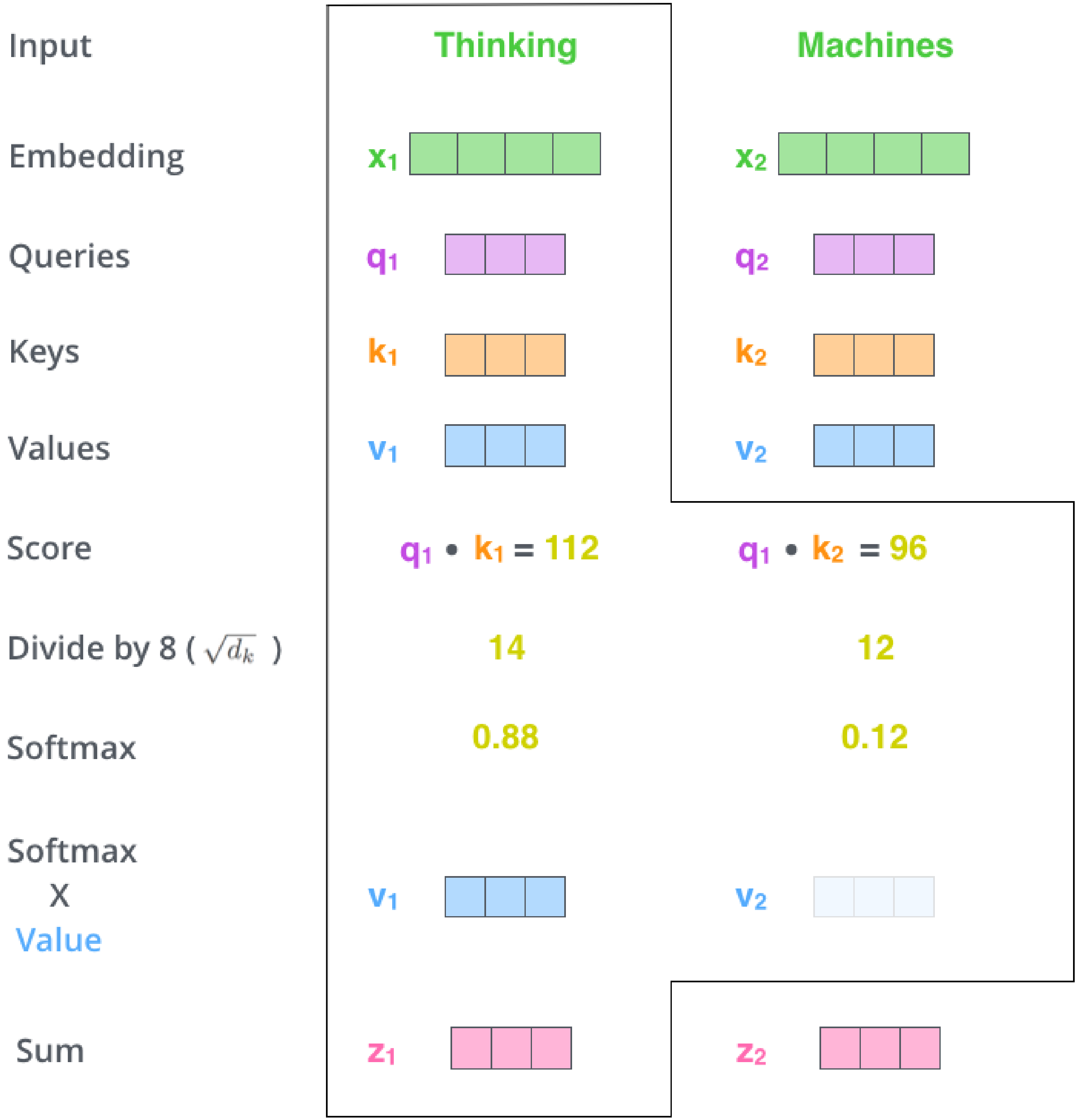


Transformer self-attention

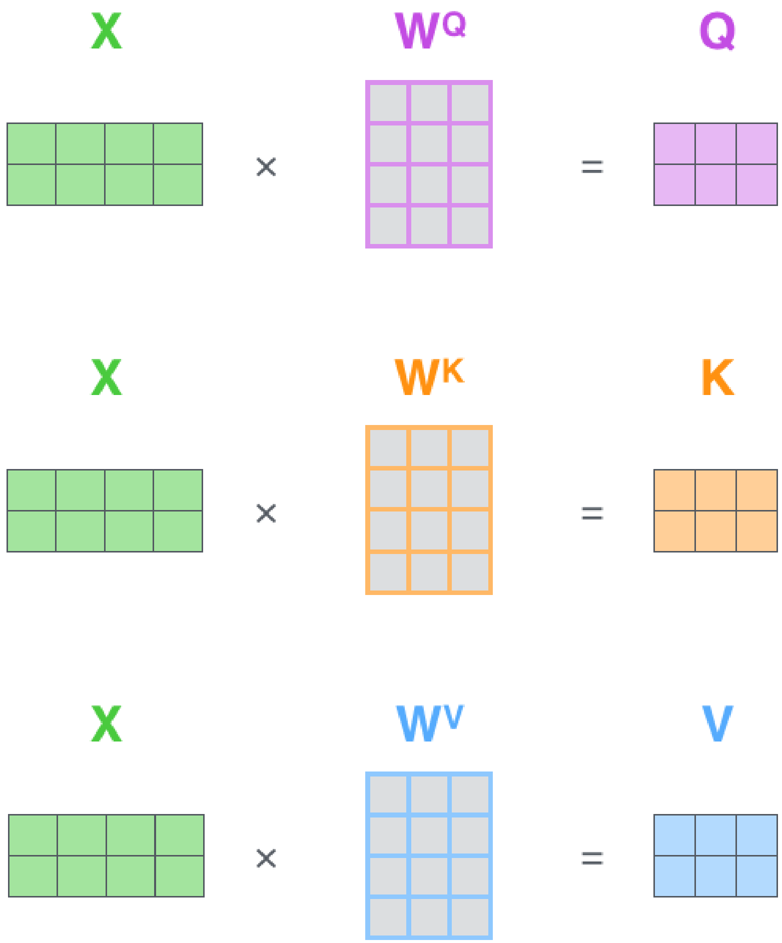


Transformer self-attention

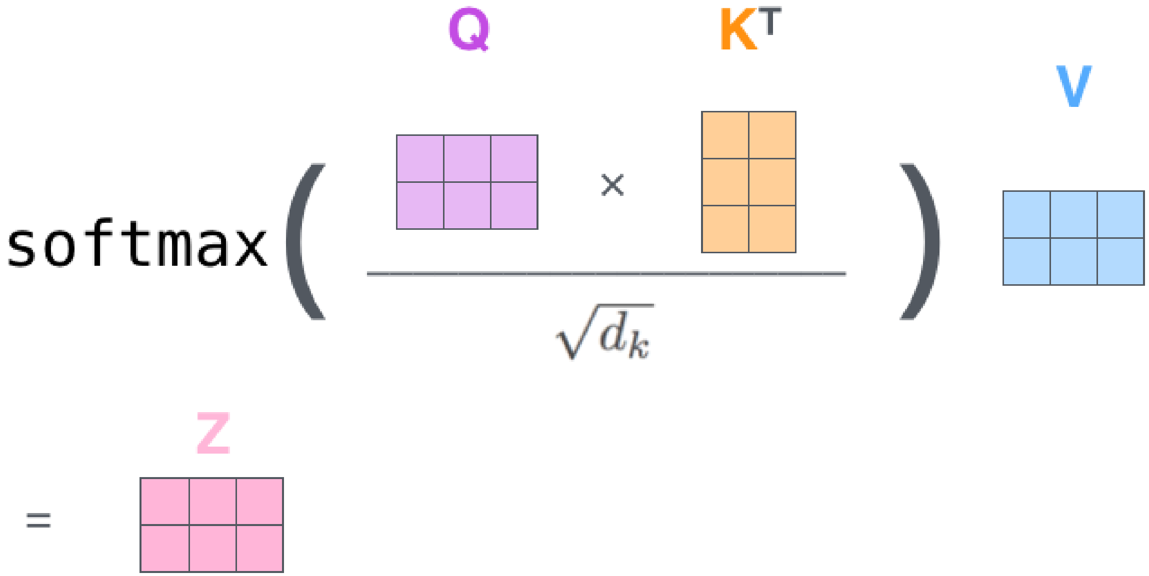



Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2

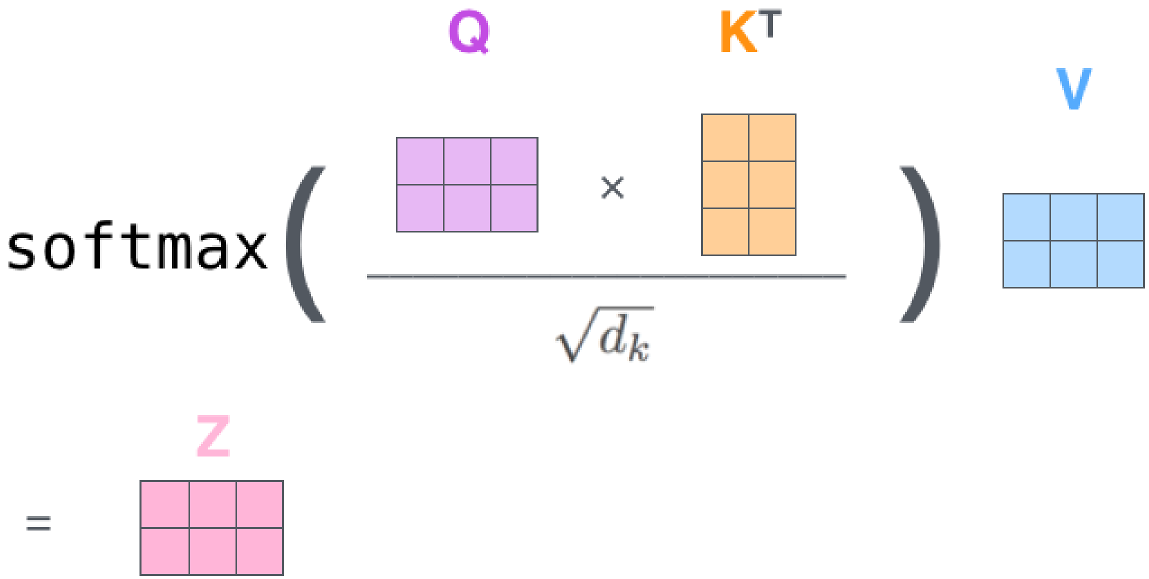
Matrix calculation



Matrix calculation



Matrix calculation



$$\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$$

$$\mathbf{Z} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\begin{cases} \mathbf{S} = \mathrm{score}(\mathbf{Q}, \mathbf{K}) \in \mathbb{R}^{n \times n} \\ \mathbf{P} = \pi(\mathbf{S}) \in \triangle^{n \times n} \\ \mathbf{Z} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{n \times d} \end{cases}$$

Problem of self-attention

- Convolution: a different linear transformation for each relative position
  - > Allows you to distinguish what information came from where

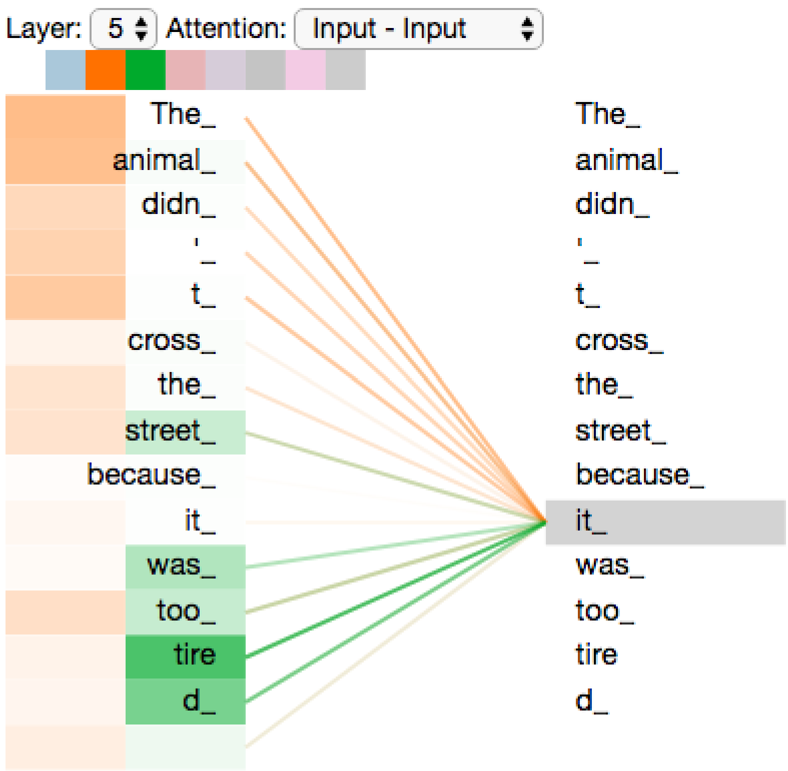
- Self-attention: a weighted average :(

Fix: multi-head attention

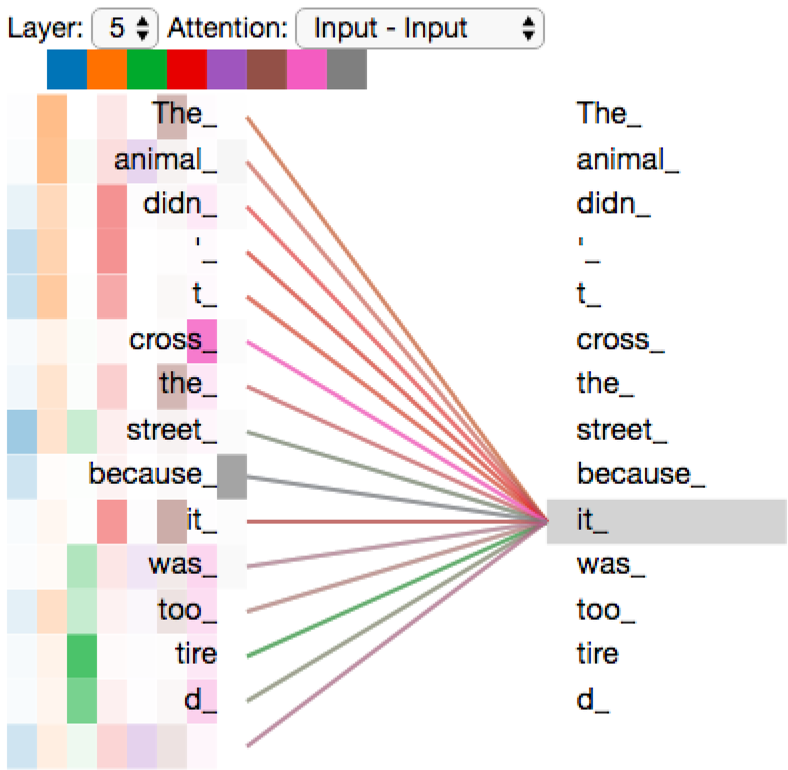
- Multiple attention layers (heads) in parallel
- Each head uses different linear transformations
- Attention layer with multiple “representation subspaces”

Multi-head attention

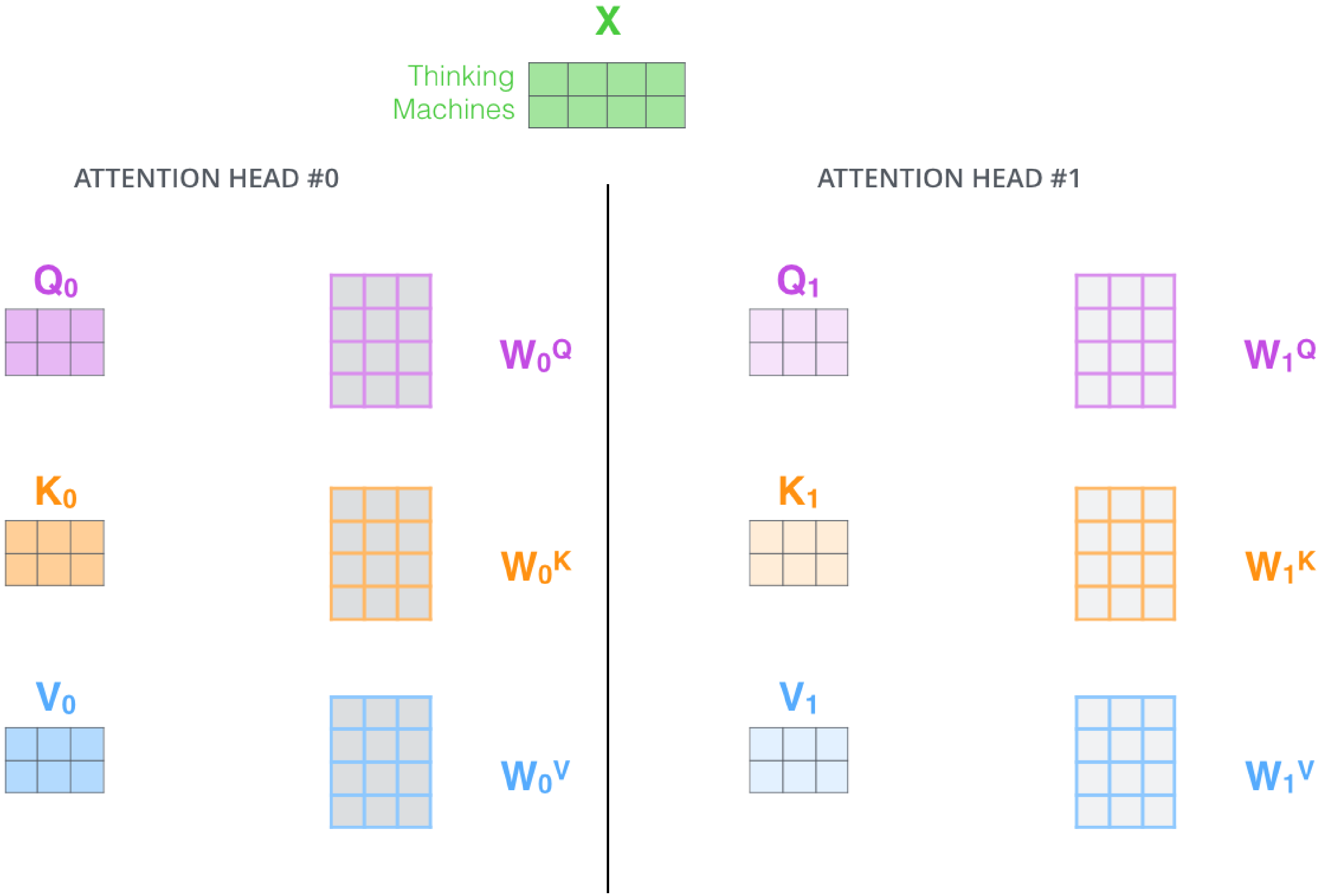
2 heads



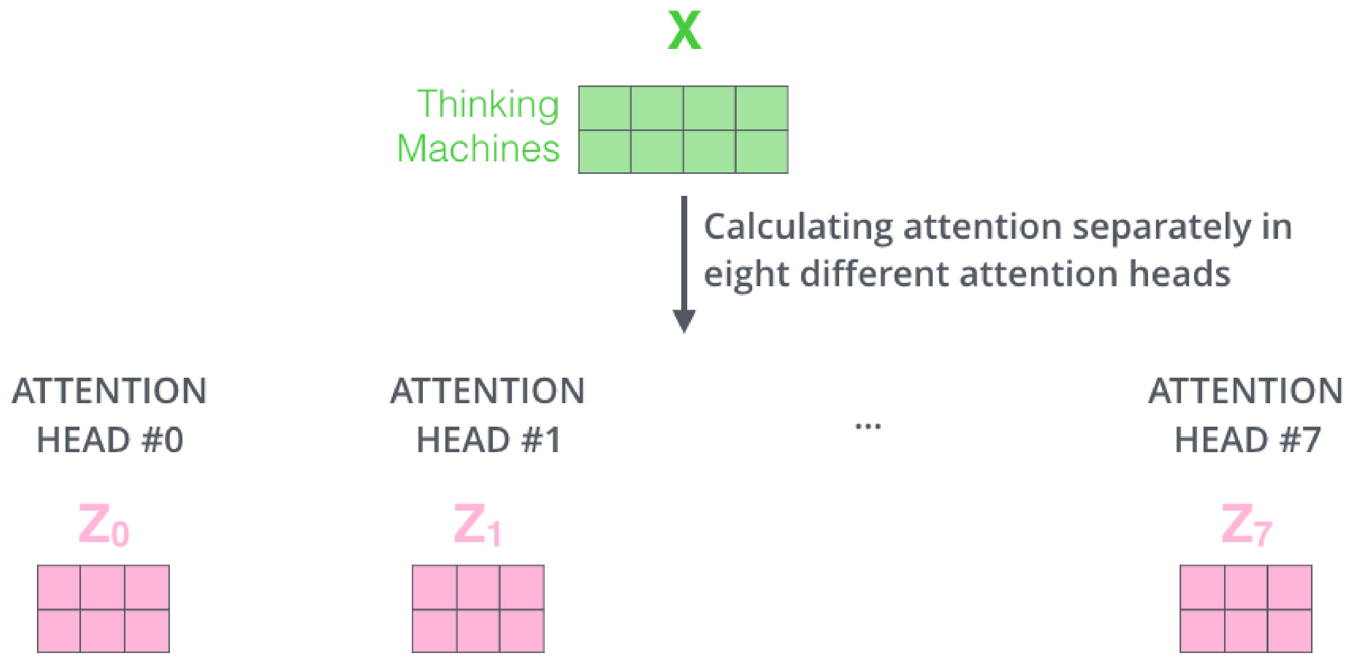
all heads (8)



Multi-head attention

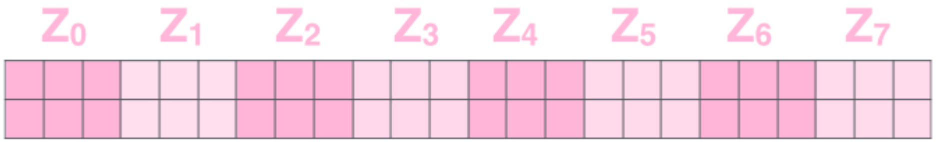


Multi-head attention



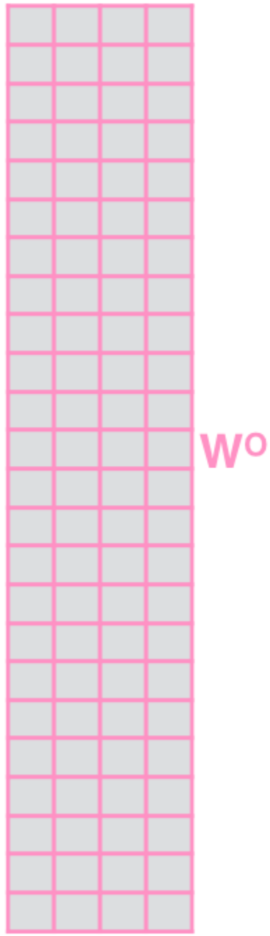
Multi-head attention

1) Concatenate all the attention heads



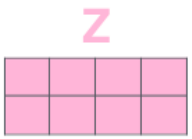
2) Multiply with a weight matrix W^O that was trained jointly with the model

x



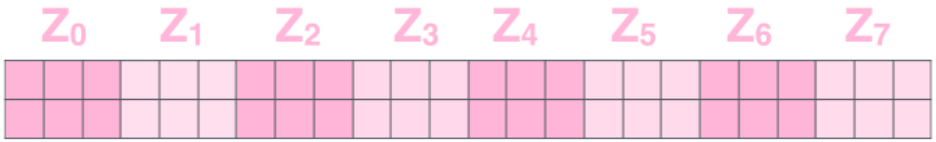
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



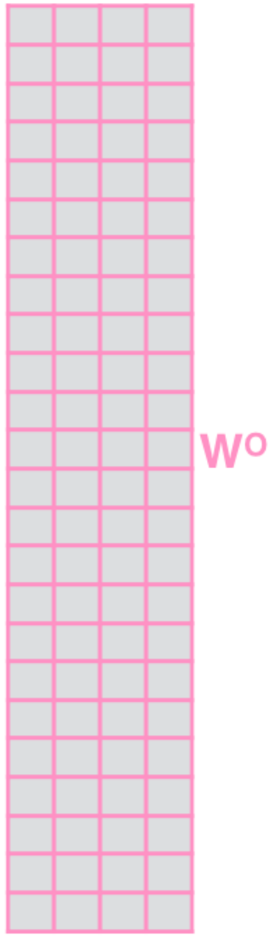
Multi-head attention

1) Concatenate all the attention heads



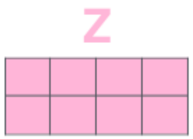
2) Multiply with a weight matrix W^O that was trained jointly with the model

x



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



$$\begin{cases} \mathbf{Z}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \\ \mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_h)\mathbf{W}^O \end{cases}$$

Multi-head attention

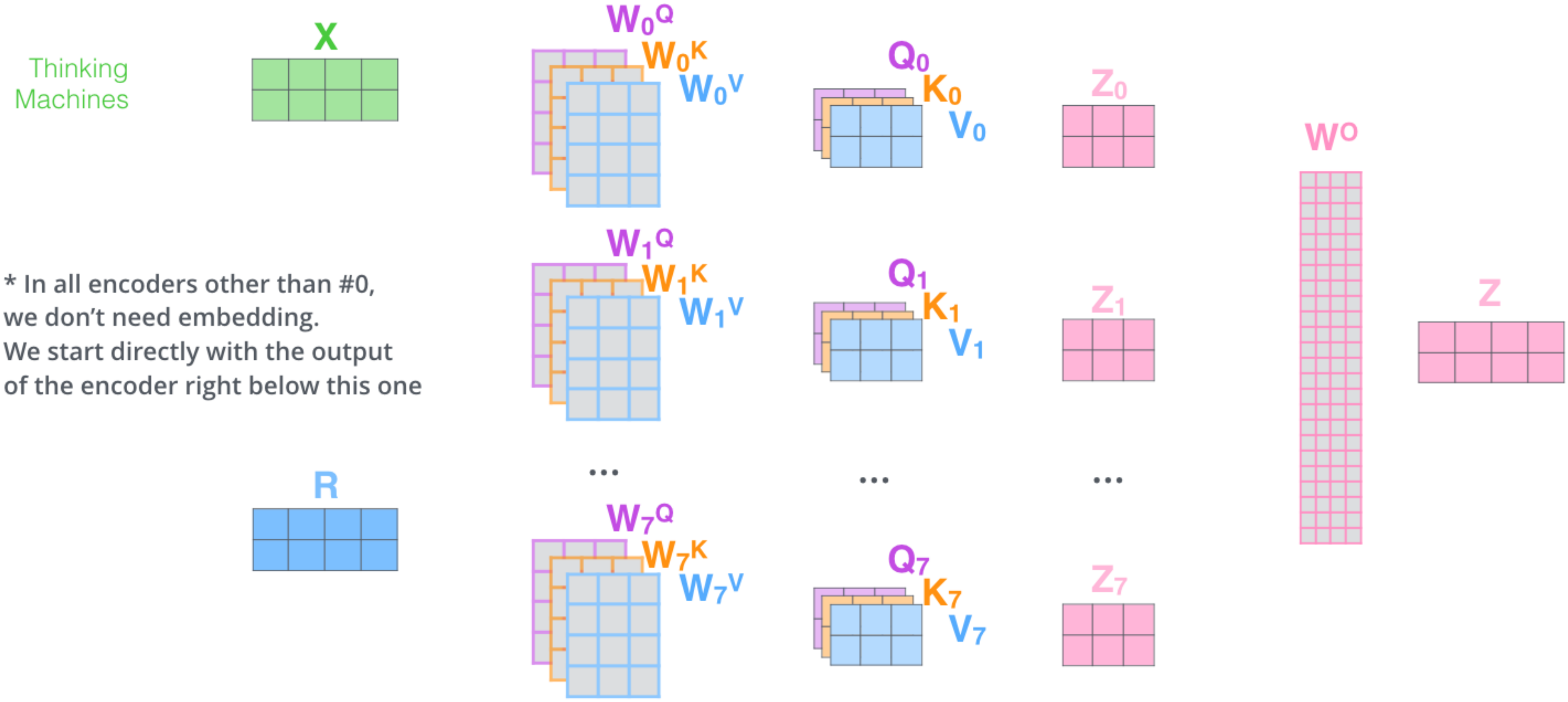
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



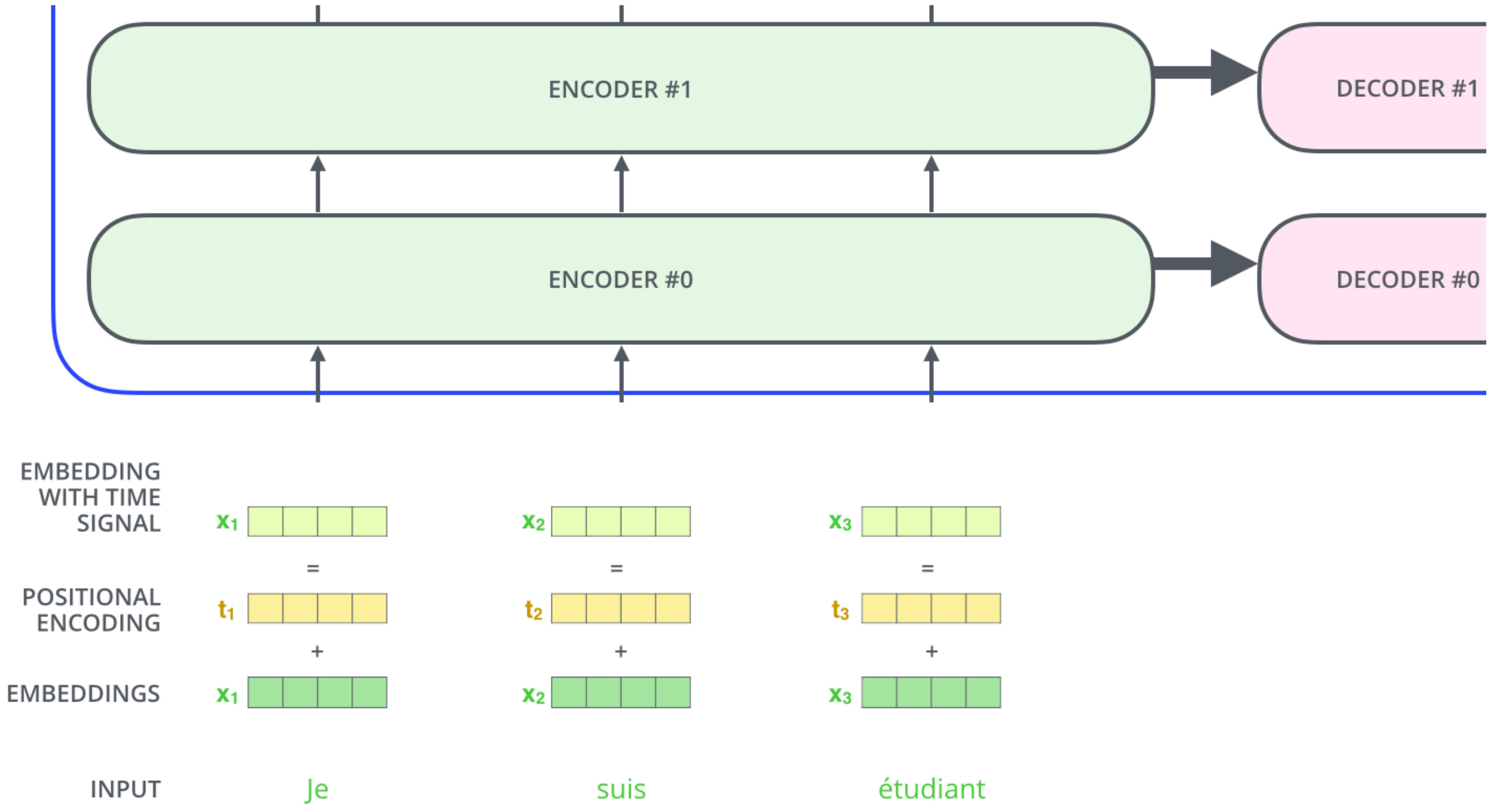
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Implementation

```
class MultiHeadAttention(nn.Module)
  def __init__(self, d_size, num_heads, dropout=0.0):
    assert d_size % num_heads == 0
    self.num_heads = num_heads
    self.h_size = d_size // num_heads
    self.linear_q = nn.Linear(d_size, self.h_size)
    self.linear_k = nn.Linear(d_size, self.h_size)
    self.linear_v = nn.Linear(d_size, self.h_size)
    self.linear_o = nn.Linear(d_size, d_size)
    self.dropout = nn.Dropout(dropout)

  def forward(self, queries, keys, values=None):
    """
    queries.shape is (batch_size, m, d)
    keys.shape is (batch_size, n, d)
    values.shape is (batch_size, n, d)
    """
    # use keys as values
    if values is None:
        values = keys

    # do all linear projections
    queries = self.linear_q(queries)
    keys = self.linear_k(keys)
    values = self.linear_v(values)

    # split heads
    batch_size = queries.shape[0]
    queries = queries.view(batch_size, -1, self.num_heads, self.h_size).transpose(1, 2)
    keys = keys.view(batch_size, -1, self.num_heads, self.h_size).transpose(1, 2)
    values = values.view(batch_size, -1, self.num_heads, self.h_size).transpose(1, 2)
```

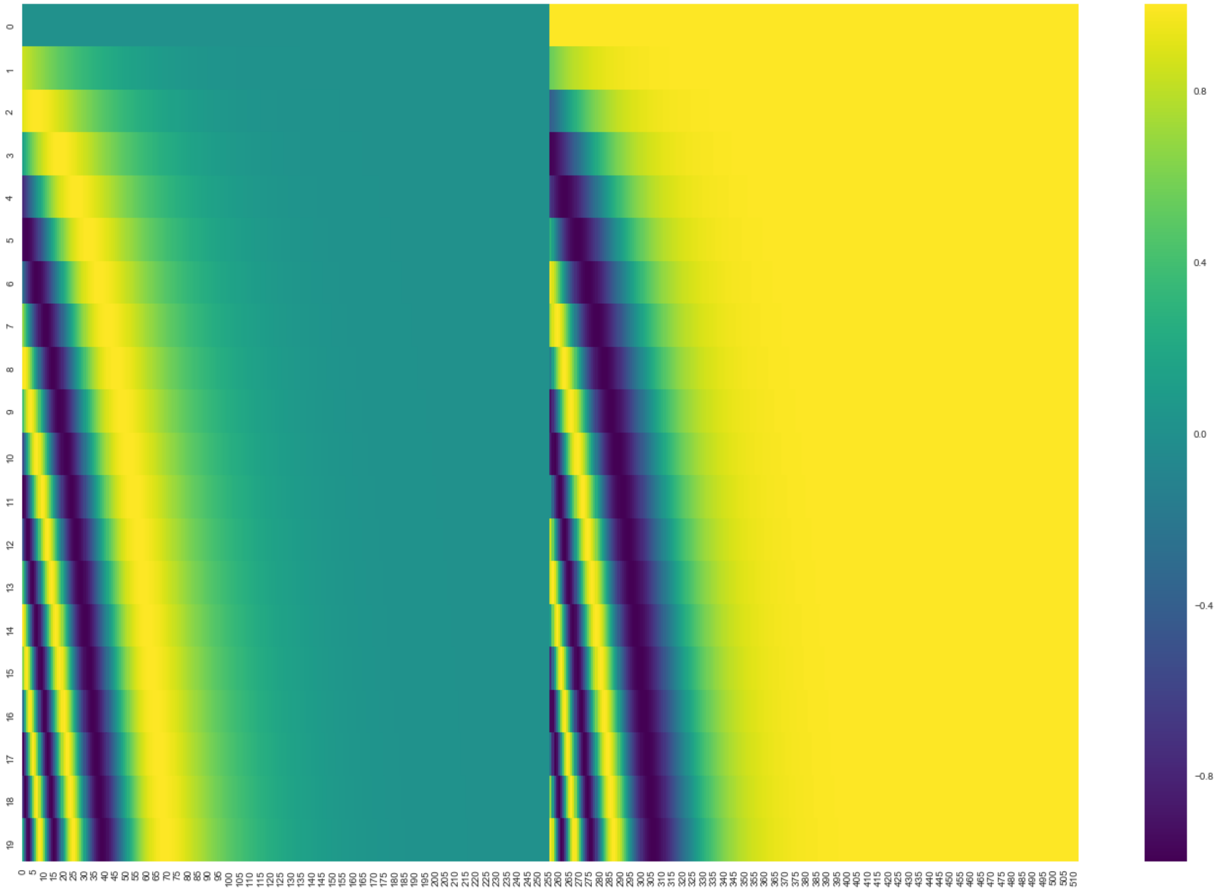
Positional encoding

- A way to account for the order of the words in the seq.



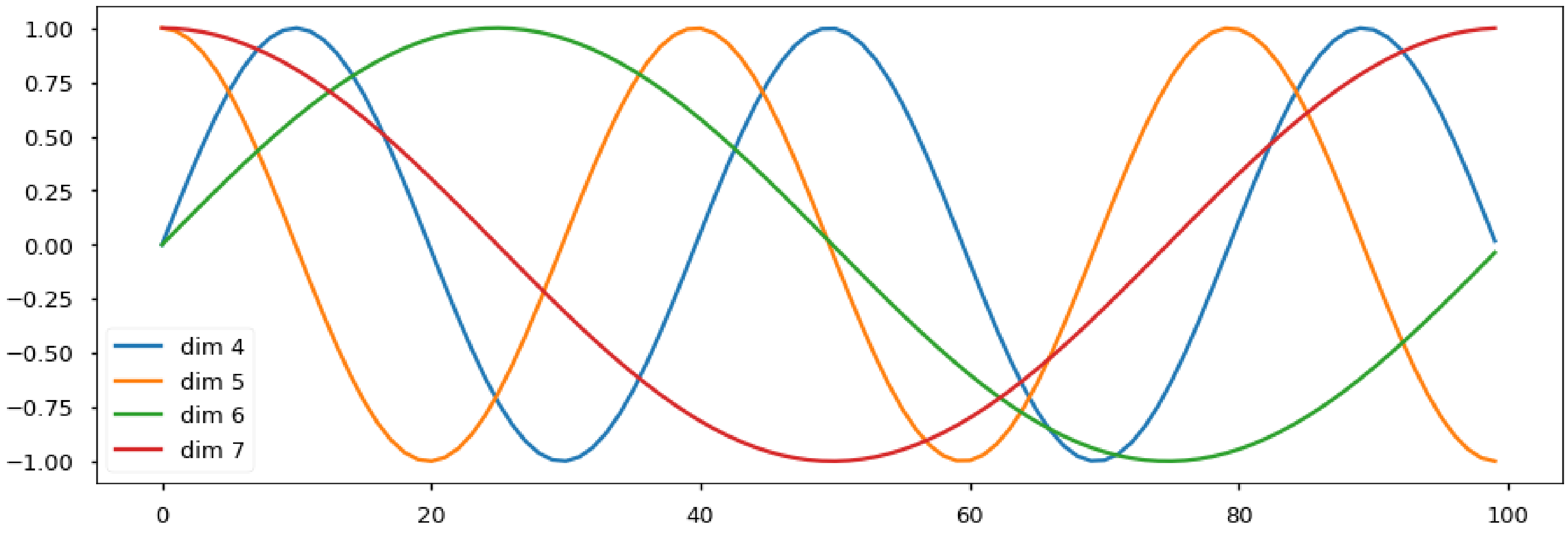
Positional encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \qquad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

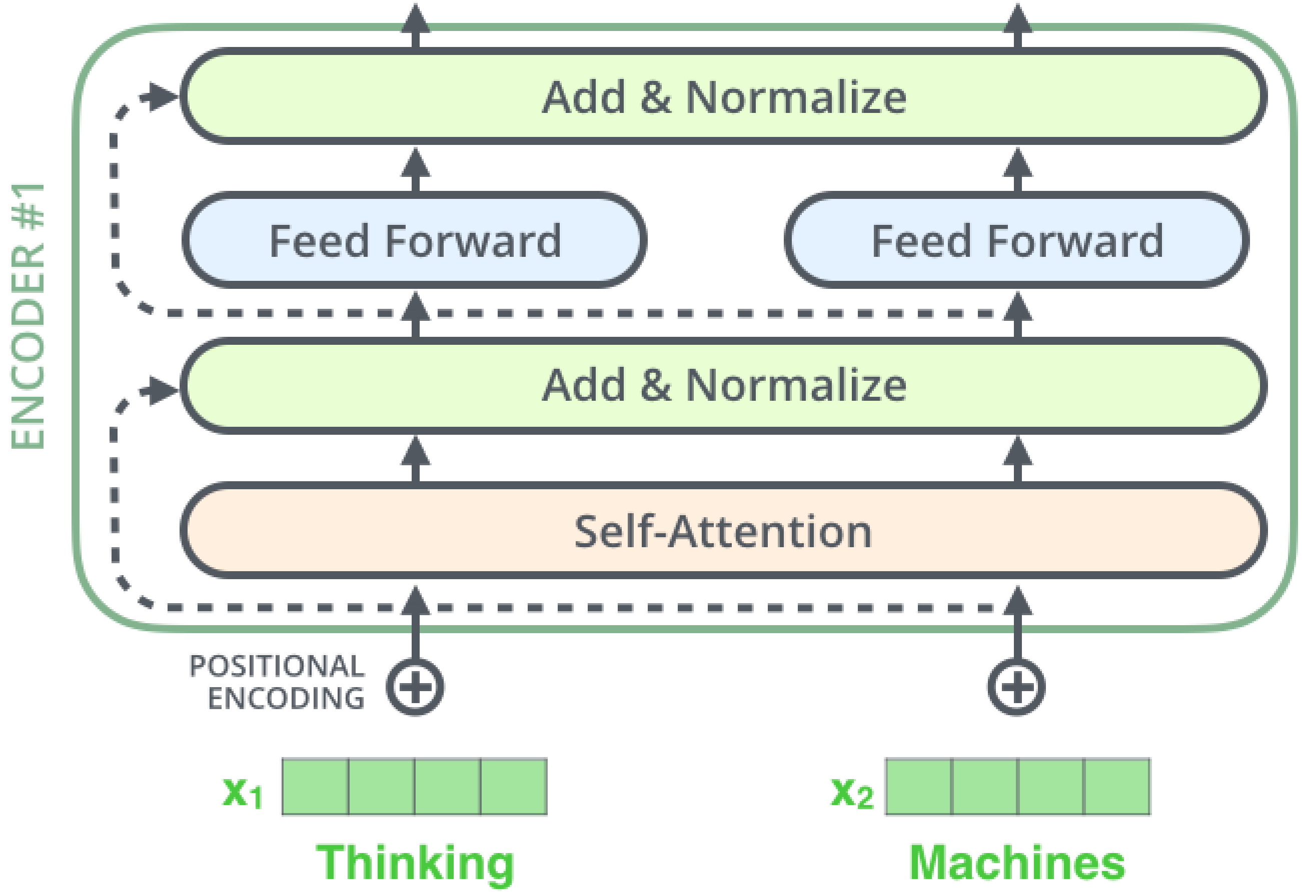


Positional encoding

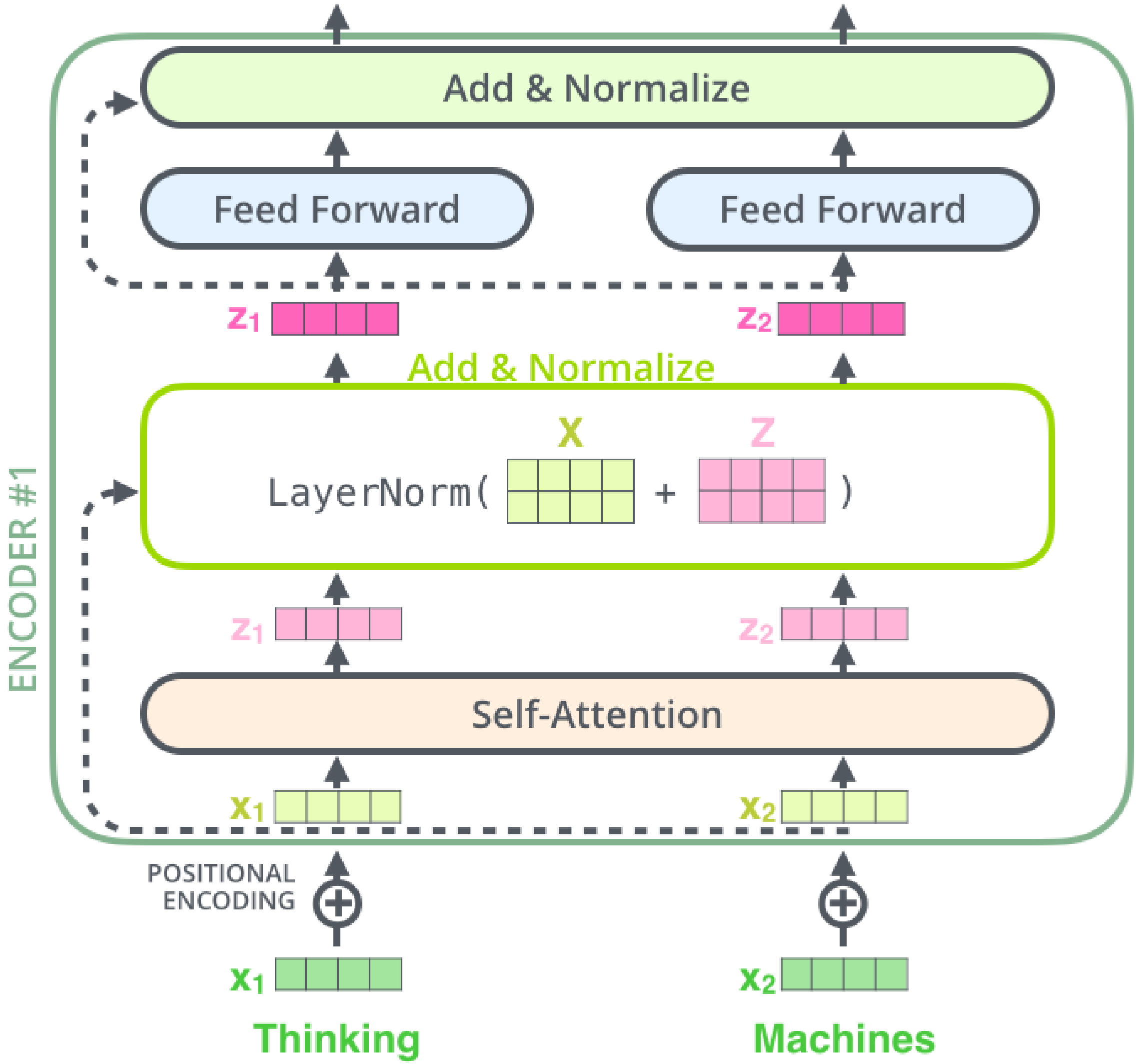
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \qquad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$



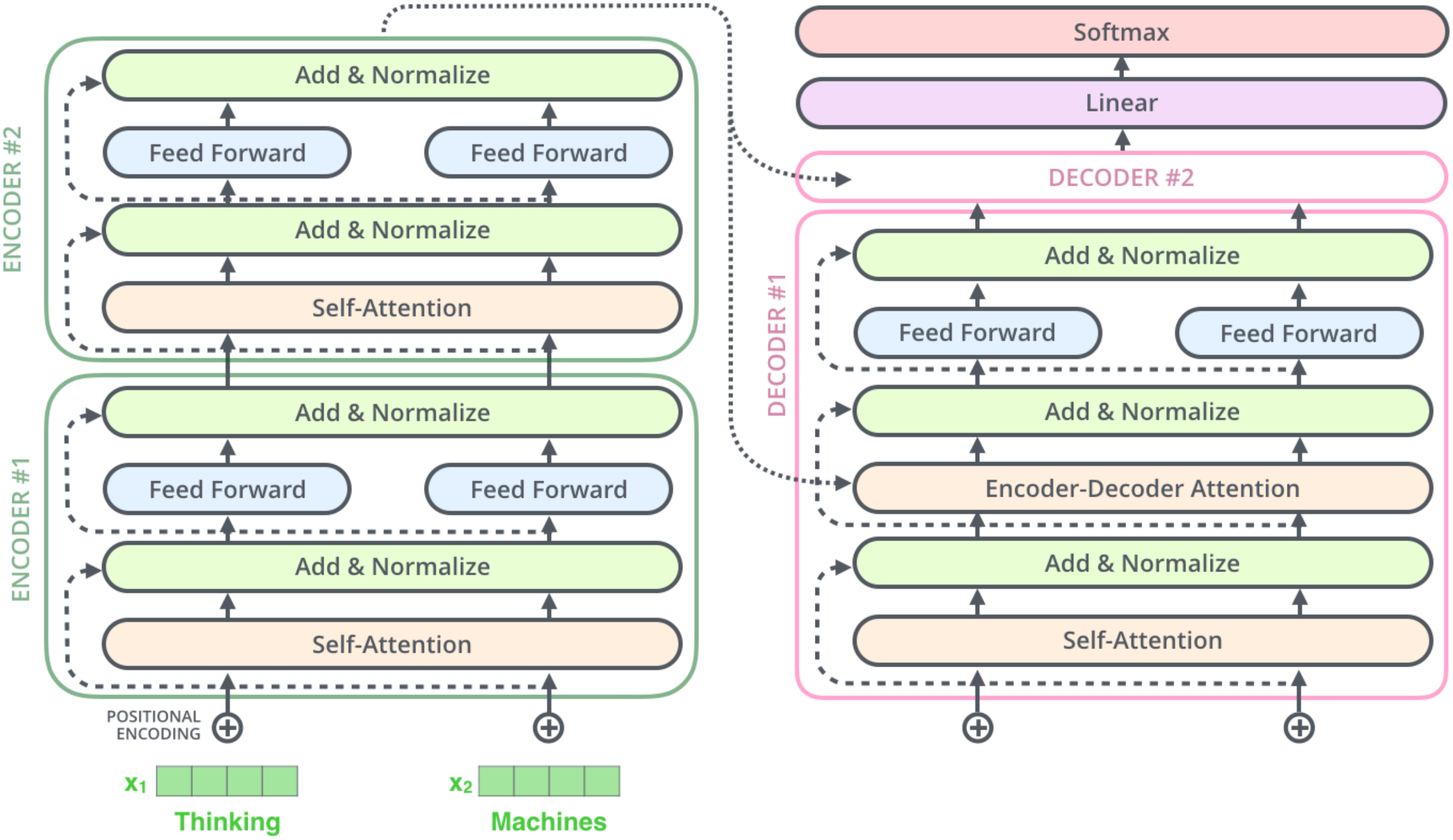
Residuals & LayerNorm



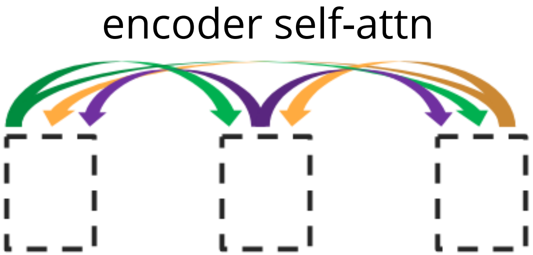
Residuals & LayerNorm



Residuals & LayerNorm

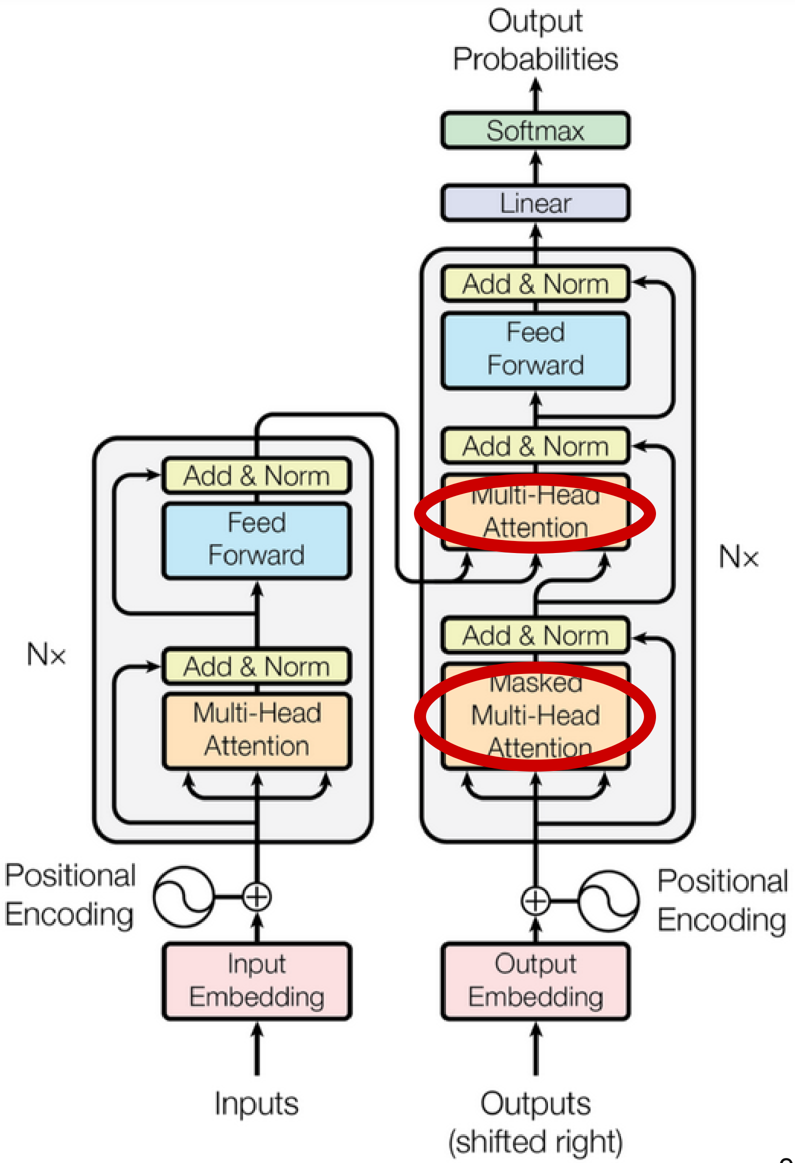


The decoder



The decoder

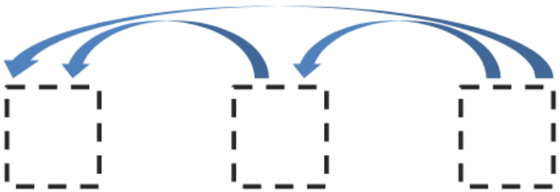
encoder self-attn

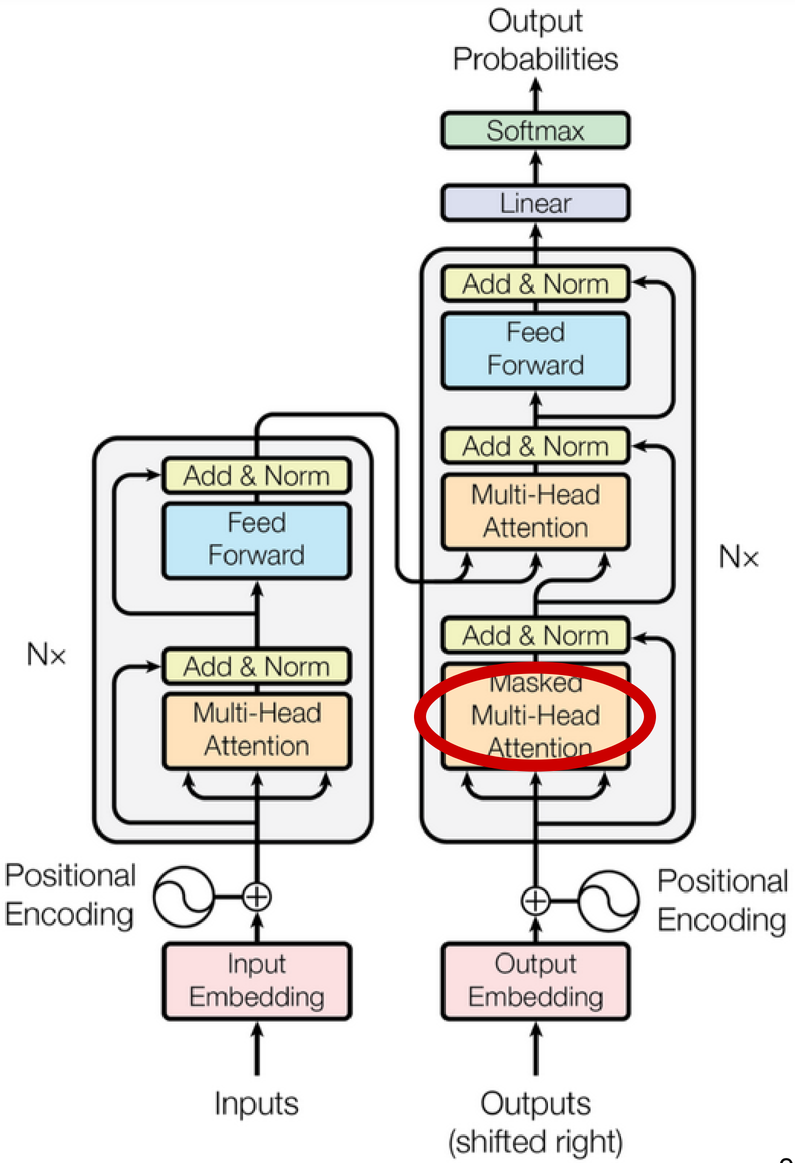


The decoder

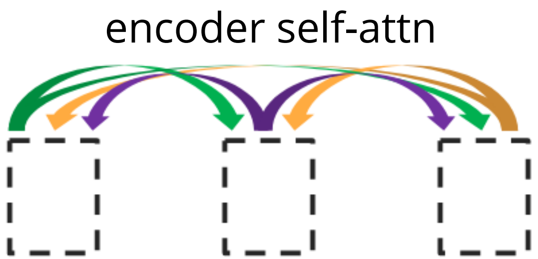
encoder self-attn

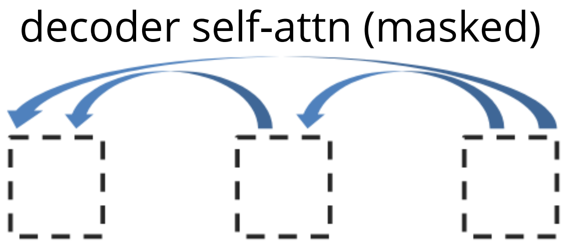
decoder self-attn (masked)



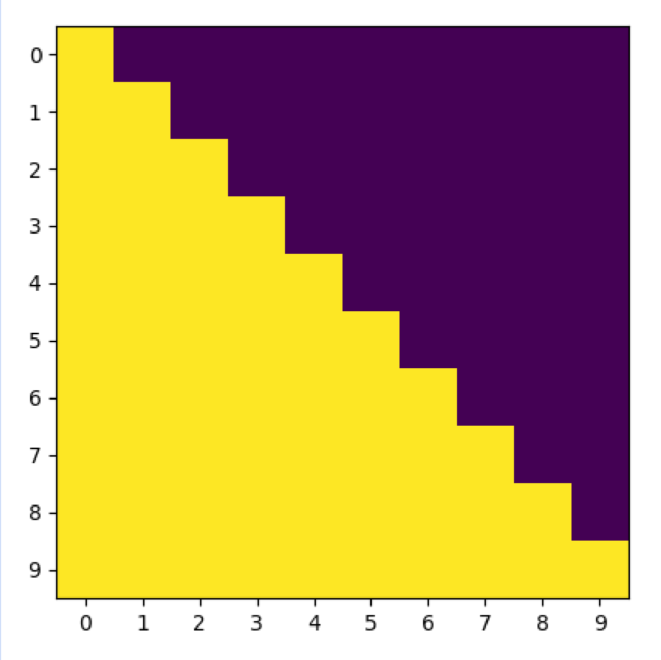


The decoder





- Mask subsequent positions (before softmax)



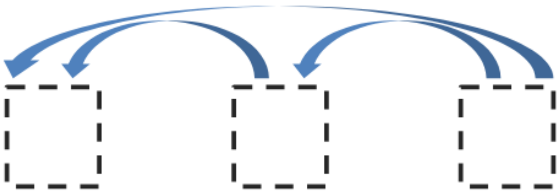
- In PyTorch

```
scores.masked_fill_(~mask, float('-inf'))
```

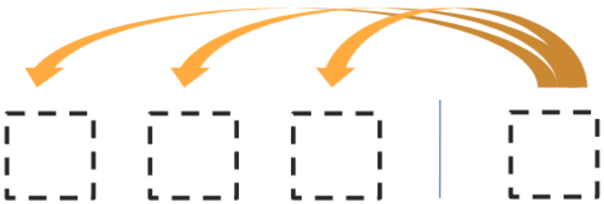
The decoder

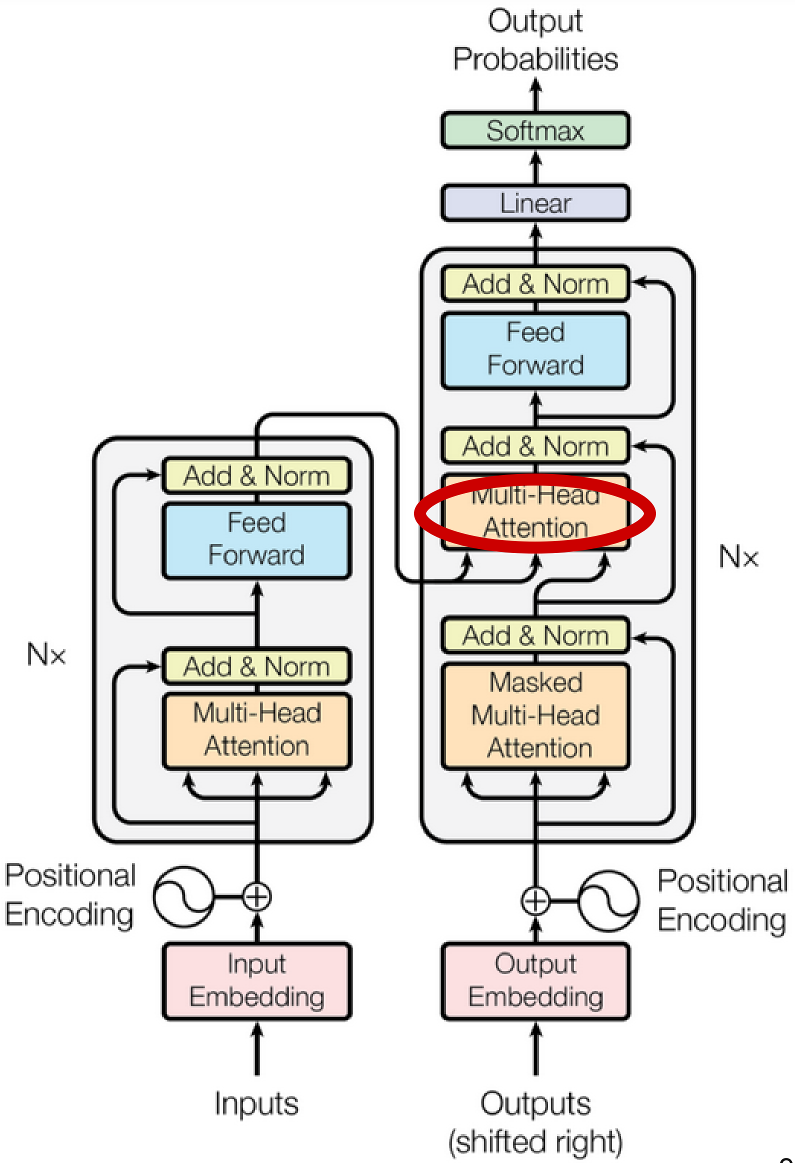
encoder self-attn

decoder self-attn (masked)

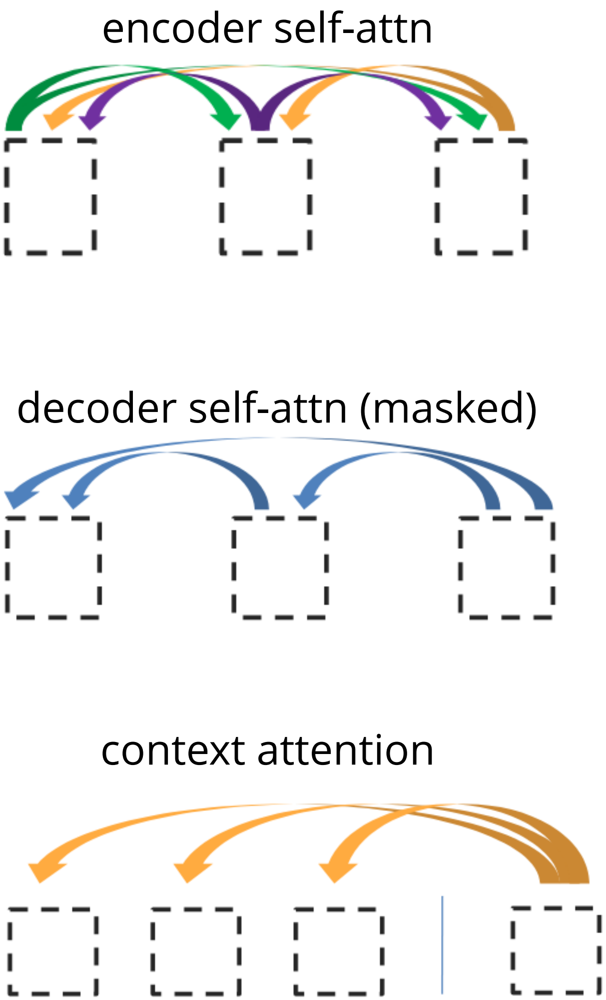


context attention





The decoder

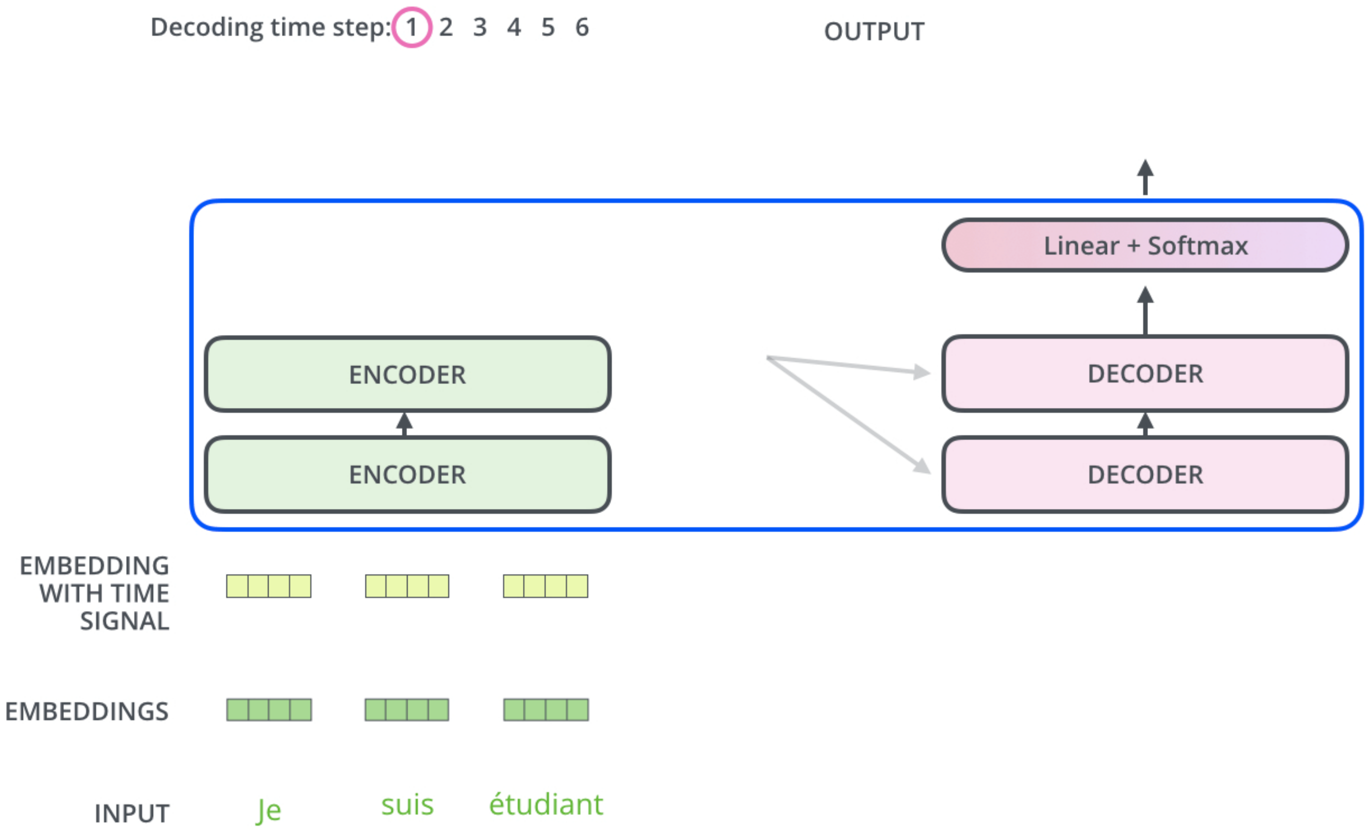


- Use the encoder output as keys and values

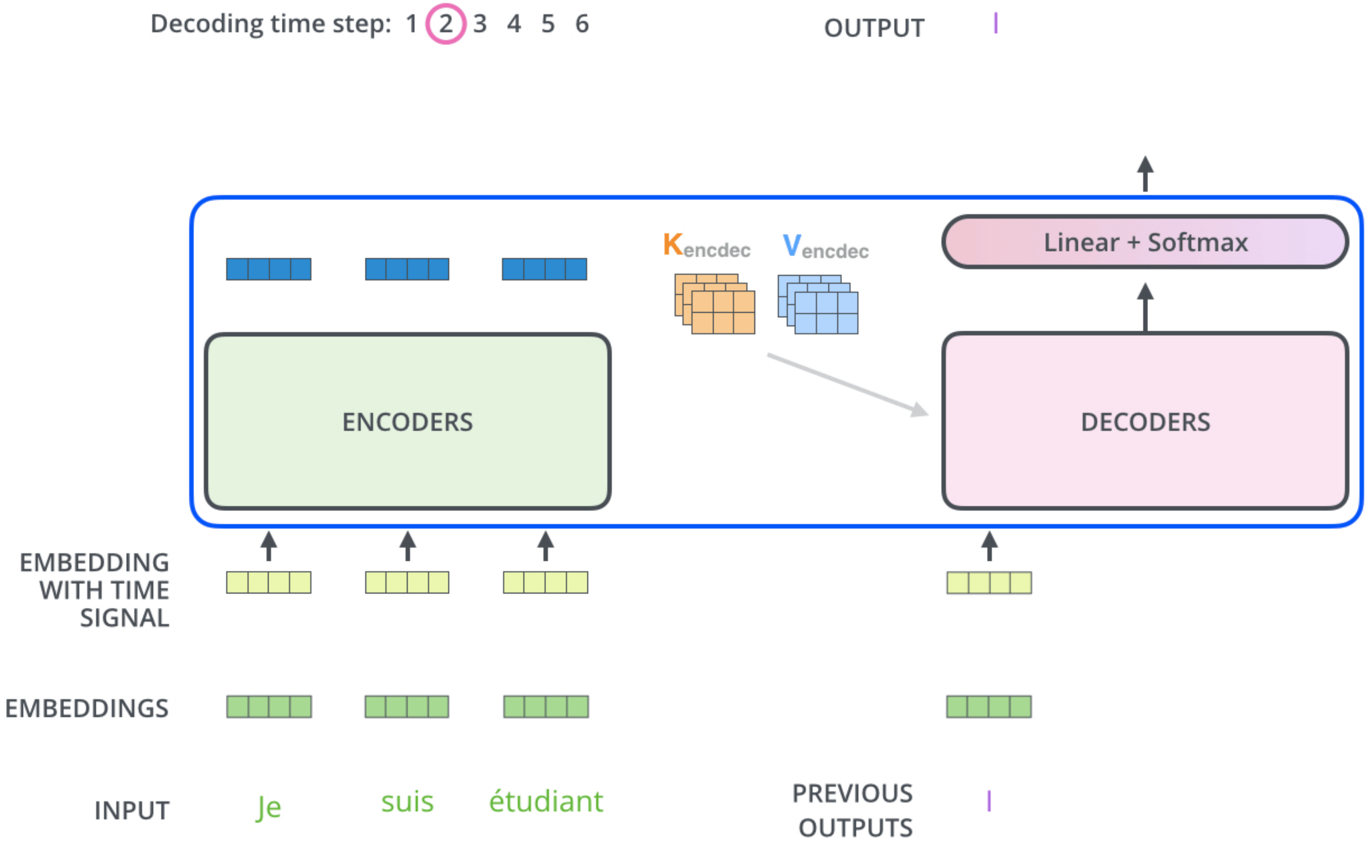
$$\mathbf{R}_{enc} = \mathrm{Encoder}(\mathbf{x}) \in \mathbb{R}^{n \times d}$$

$$\begin{cases} \mathbf{S} = \mathrm{score}(\mathbf{Q}, \mathbf{R}_{enc}) \in \mathbb{R}^{m \times n} \\ \mathbf{P} = \pi(\mathbf{S}) \in \triangle^{m \times n} \\ \mathbf{Z} = \mathbf{P}\mathbf{R}_{enc} \in \mathbb{R}^{m \times d} \end{cases}$$

The decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

The decoder



Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

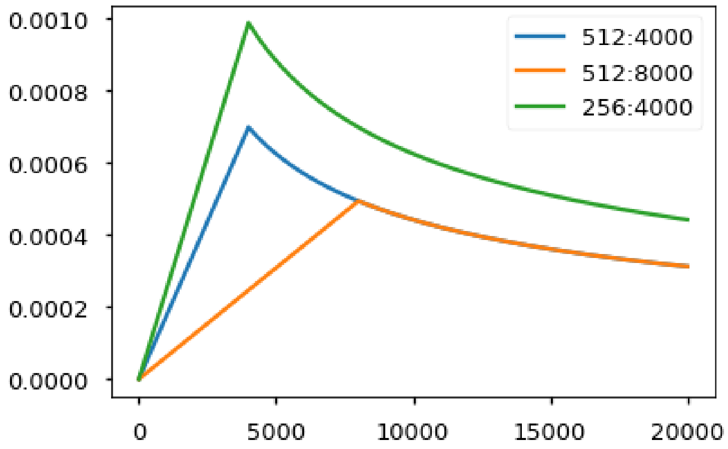
Computational cost

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n = seq. length $\quad$ d = hidden dim $\quad$ k = kernel size

Other tricks 🔮

- Training Transformers is like black-magic. There are a bunch of other tricks:
  - Label smoothing
  - Dropout at every layer before residuals
  - Beam search with length penalty
  - Subword units - BPEs
  - Adam optimizer with **learning-rate decay**



Replacing recurrence

- Self-attention is the only place where positions interact

- What do we gain over RNN-based models?

- What do we lose?

Coding & training tips

- Sasha Rush's post is a **really good** starting point:

  http://nlp.seas.harvard.edu/2018/04/03/attention.html

- OpenNMT-py implementation:

  encoder part | decoder part

  on the "good" order of LayerNorm and Residuals

- PyTorch has a built-in implementation since August, 2019

  torch.nn.Transformer

- Training Tips for the Transformer Model

  https://arxiv.org/pdf/1804.00247

What else?

- BERT uses only the encoder side (Devlin et al., 2018)
- GPT-3 uses only the decoder side (Brown et al., 2020)

- Absolute vs relative positional encoding (Shaw et al., 2018)

- Use previous encoded states as memory
  - Transformer-XL (Dai et al., 2019)
  - Compressive Transformer (Rae et al., 2019)

- Induce sparsity
  - Sparse Transformer (Child et al., 2019)
  - Span Transformer (Sukhbaatar et al., 2019)
  - Adap. Sparse Transformer (Correia et al., 2019)

learn an α in entmax for each head:

$$\frac{\partial \alpha\text{-}\mathrm{entmax}(\boldsymbol{\theta})}{\partial \alpha}$$

Subquadratic self-attention

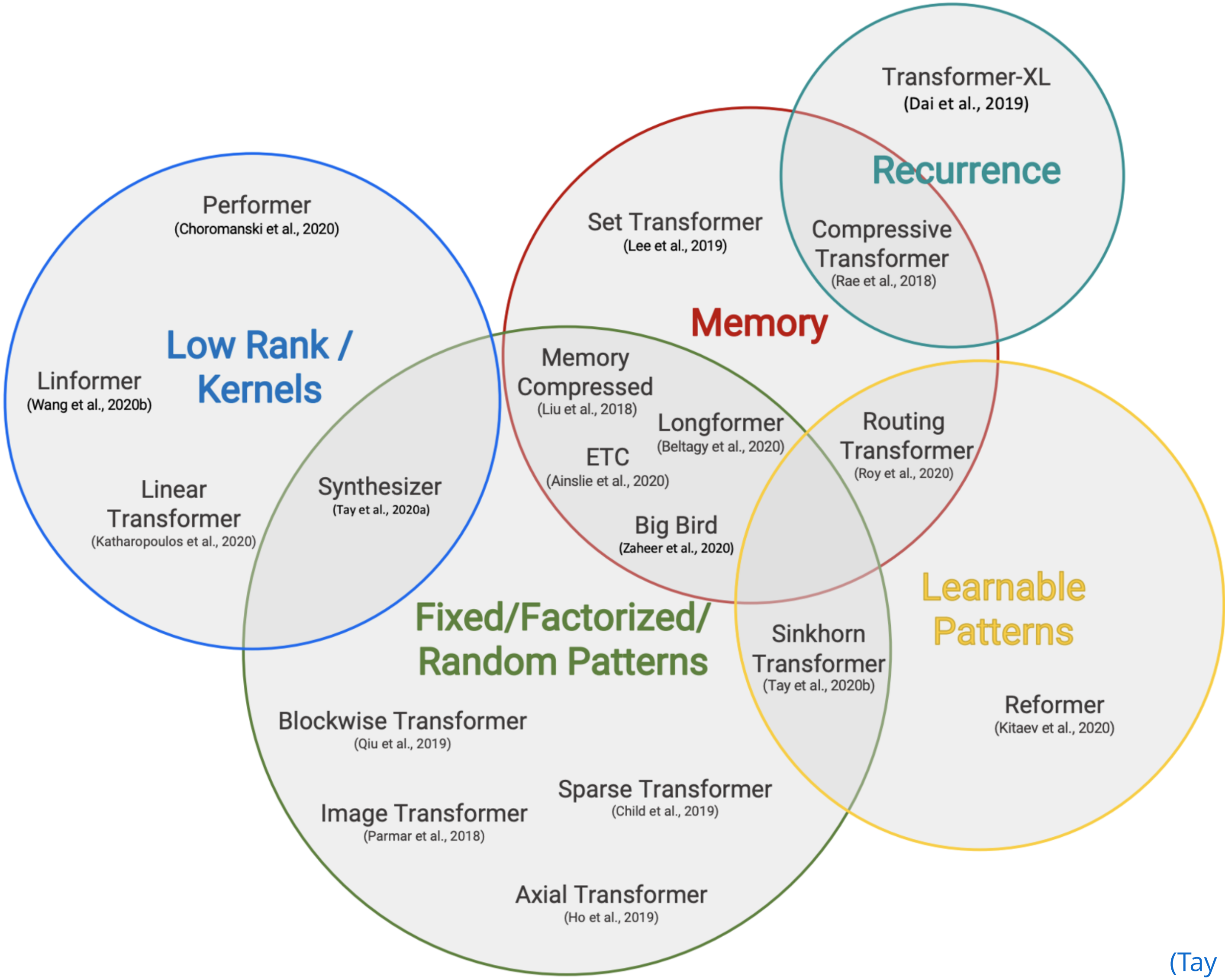
🔥 very active research topic! why?

$O(n^2) \quad \ldots \quad O(n \log n) \quad \ldots \quad O(n)$

+🚀 −💾 $\implies$ −💰

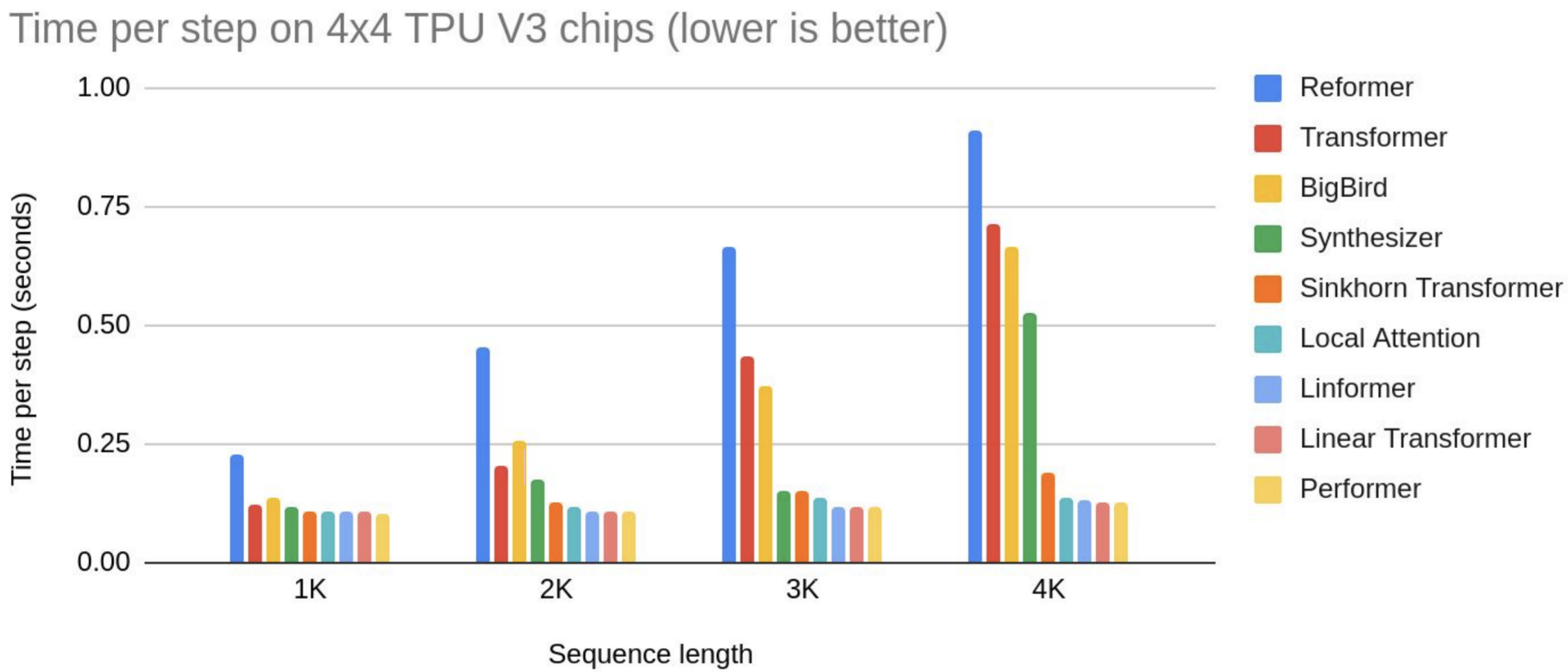
+🚀 −💾 $\implies$ +🌱

Subquadratic self-attention



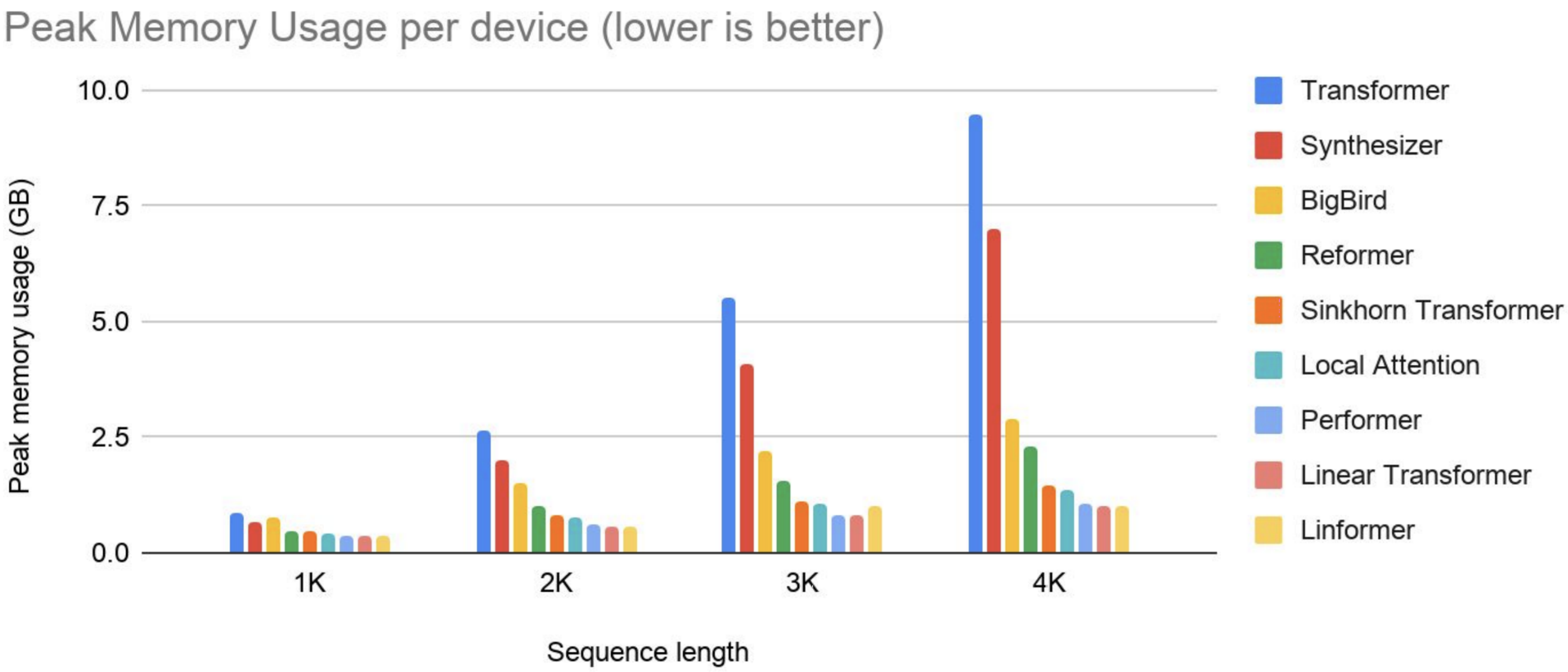
(Tay et al., 2020)

Subquadratic self-attention



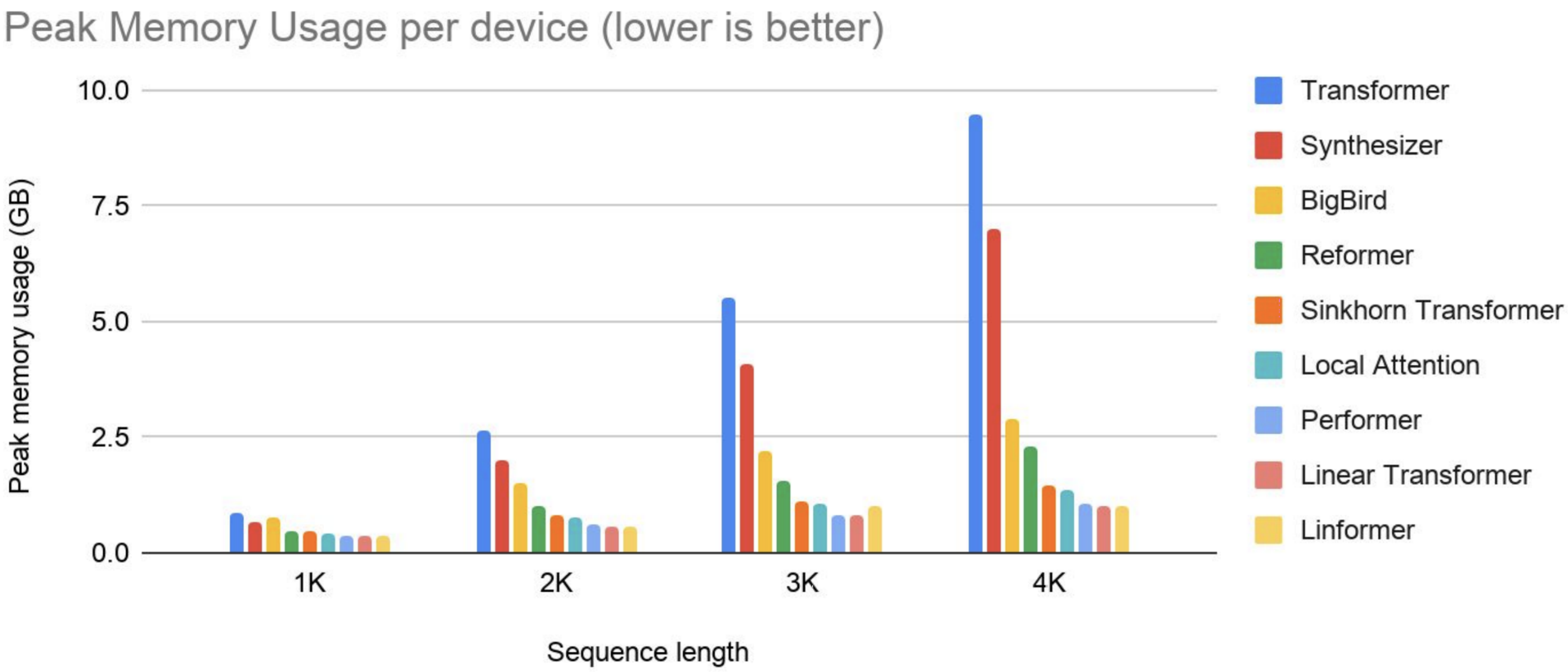
(Ilharco et al., 2020)

Subquadratic self-attention



(Ilharco et al., 2020)

Subquadratic self-attention



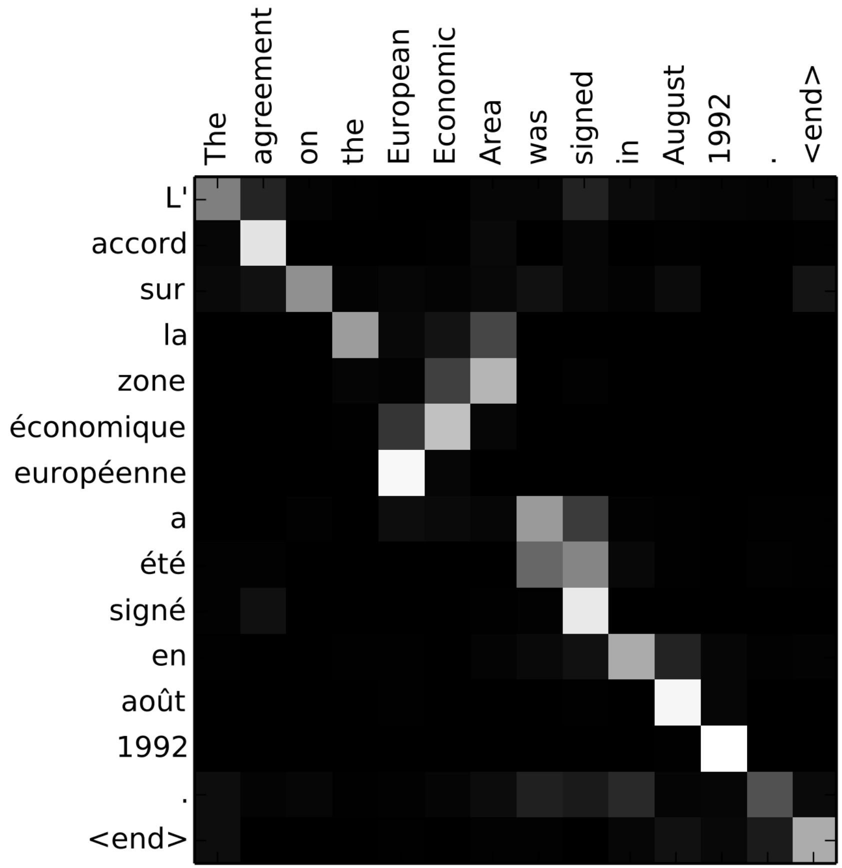
High Performance NLP - EMNLP 2020 (Ilharco et al., 2020)

Pause

🚰

Attention interpretability

🦕 an early example in NLP: *alignments* ⇄ *attention*



(Bahdanau et al., 2015)

Attention interpretability

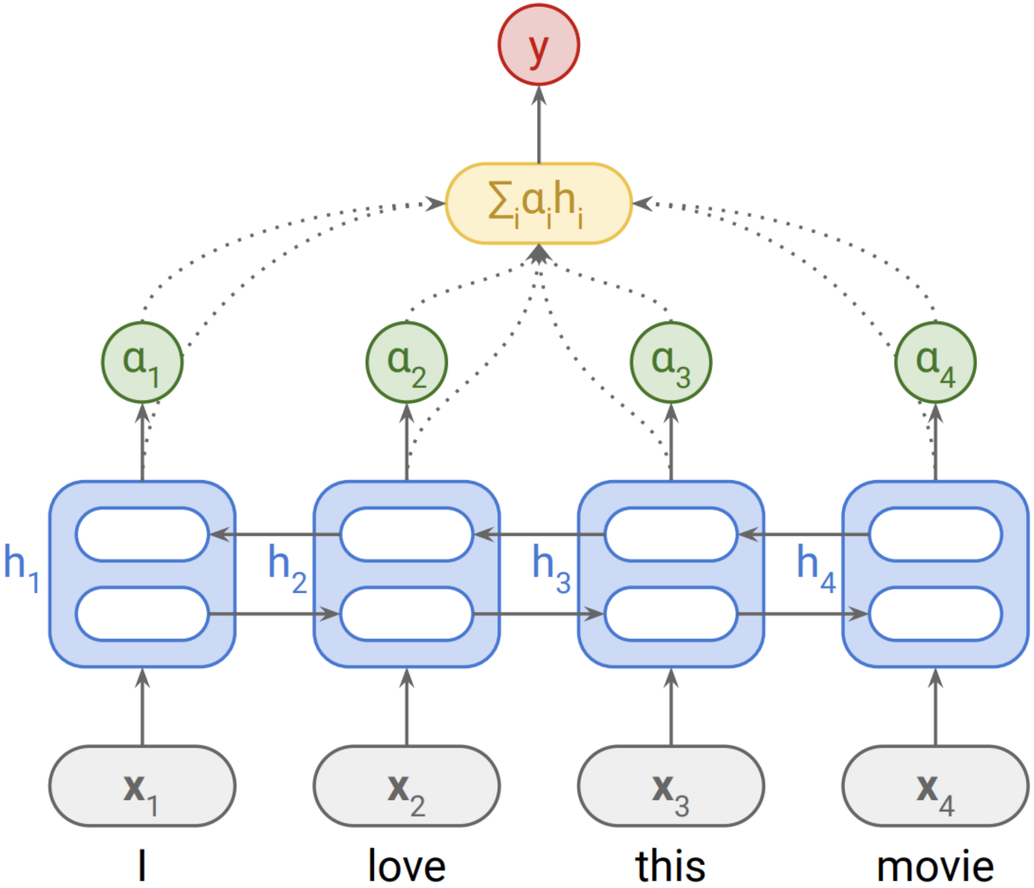
- What is explainability? interpretability? transparency?
  - See this recent work: (Verma et al., 2020)
  - See Explainable AI Tutorial - AAAI 2020

- What is the overall goal of the explanation by attention?

expose which tokens are the most important ones for a particular prediction => saliency map

- To whom are we explaining?
  - 👶👶 Non-experts
  - 👩‍💼👨‍💼 Investors
  - 👩‍💻👨‍💻 **Model developers**

Attention debate



BiLSTM with attention - basic architecture for text classification tasks

(Bastings and Filippova, 2020)

Attention is not explanation

- Do attention weights correlate with gradient and leave-one-out measures?

Gradient:

$$\nabla_{\mathbf{x}_i} f(\mathbf{x}_{1:n}) \cdot \mathbf{x}_i$$

- First-order Taylor expansion near $\mathbf{x}_i$
- Linear model: gradient=coefficients

Leave-one-out:

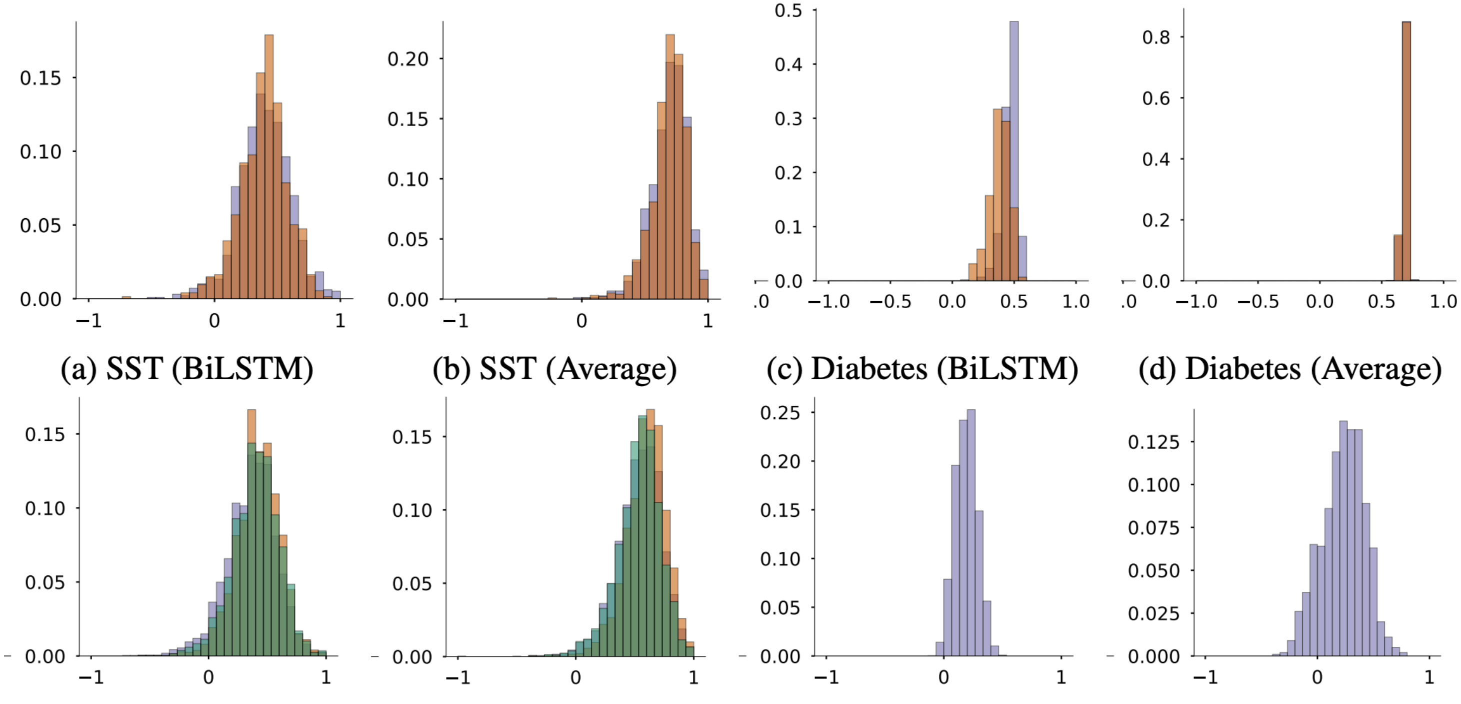
$$f(\mathbf{x}_{1:n}) - f(\mathbf{x}_{-i})$$

- First-order Taylor expansion near $\mathbf{x}_i$
- Linear model: gradient=coefficients

(Jain and Wallace, 2019)

Attention is not explanation

- Do attention weights correlate with gradient and leave-one-out measures? **No!**



(a) SST (BiLSTM) (b) SST (Average) (c) Diabetes (BiLSTM) (d) Diabetes (Average)

(e) SNLI (BiLSTM) (f) SNLI (Average) (g) CNN-QA (BiLSTM) (h) BAbI 1 (BiLSTM)

(Jain and Wallace, 2019)

Attention is not explanation

- Do attention weights correlate with gradient and leave-one-out measures? **No!**

- Can we find alternative attention distributions $\tilde{\alpha}$ that yield the same prediction as the original $\alpha^{\star}$?

Adversarial attention:

$$\max_{\tilde{\alpha} \in \triangle^n} f_{\tilde{\alpha}}(\mathbf{x}_{1:n})$$

$$\text{s.t.} \quad |f_{\tilde{\alpha}(\mathbf{x}_{1:n})} - f_{\alpha^{\star}}(\mathbf{x}_{1:n})| < \epsilon$$

(Jain and Wallace, 2019)

Attention is not explanation

- Do attention weights correlate with gradient and leave-one-out measures? **No!**

- Can we find alternative attention distributions $\tilde{\alpha}$ that yield the same prediction as the original $\alpha^\star$? **Yes! Easily!**

after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

original α

$$f(x|\alpha, \theta) = 0.01$$

after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

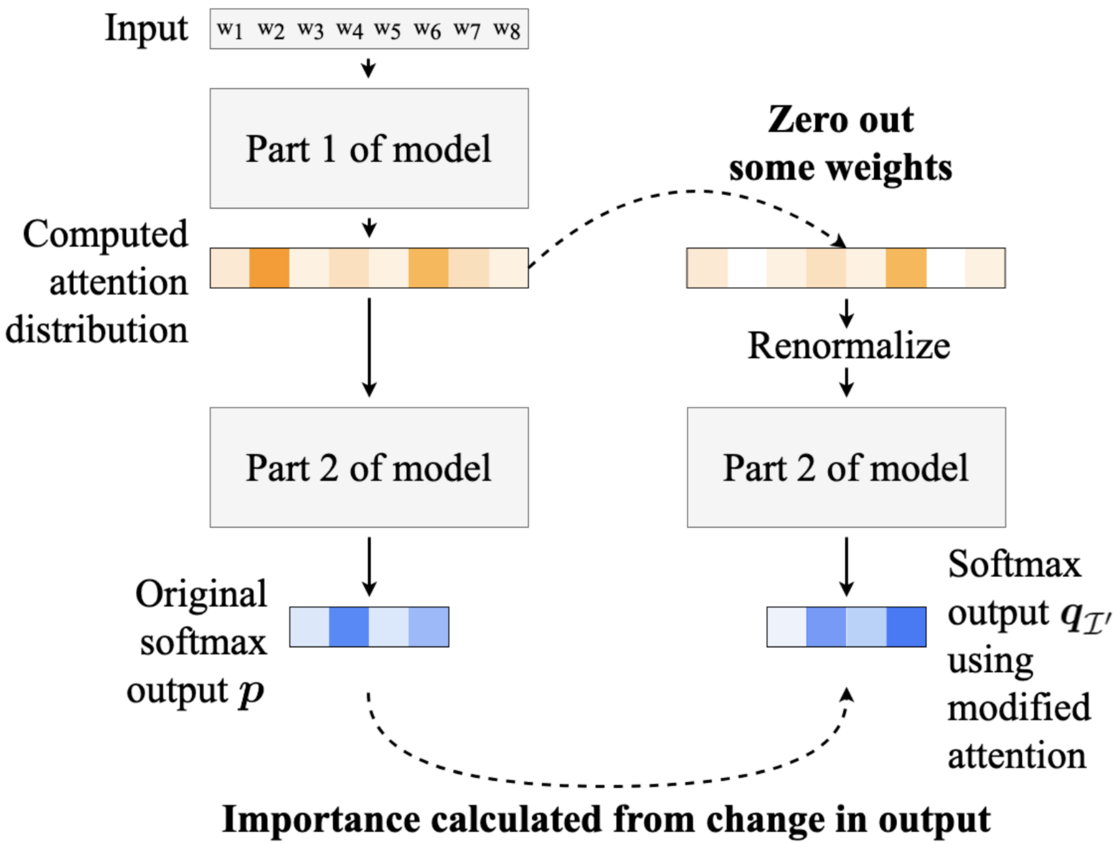
adversarial $\tilde{\alpha}$

$$f(x|\tilde{\alpha}, \theta) = 0.01$$

(Jain and Wallace, 2019)

Is attention interpretable?

- What happens if we erase the highest attention weight and re-normalize the distribution? Does the decision flips?

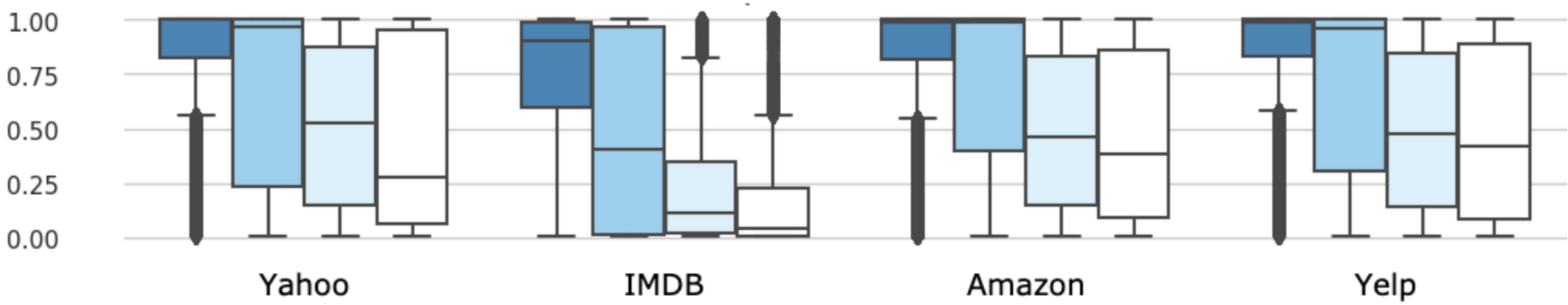


(Serrano and Smith, 2019)

Is attention interpretable?

- What happens if we erase the highest attention weight and re-normalize the distribution? Does the decision flips? **No!**

Fraction of original attention weights removed before first decision flip

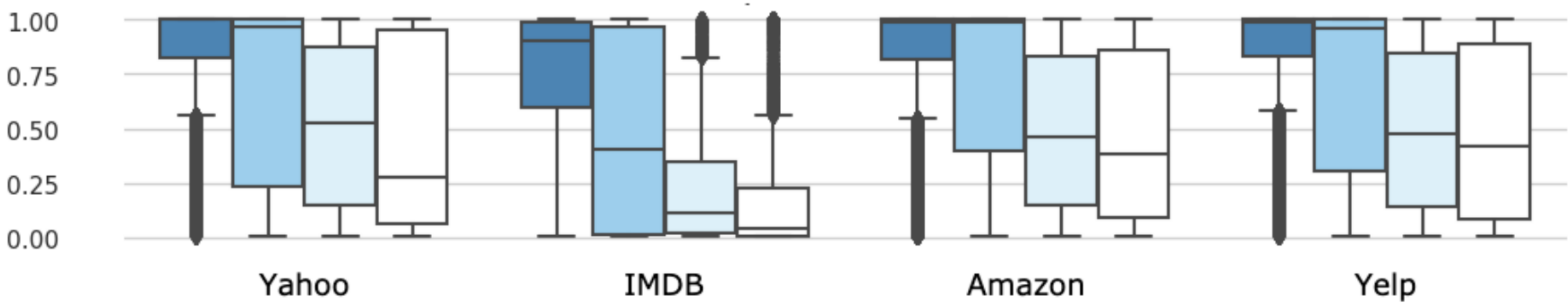


(Serrano and Smith, 2019)

Is attention interpretable?

- What happens if we erase the highest attention weight and re-normalize the distribution? Does the decision flips? **No!**

Fraction of original attention weights removed before first decision flip



"the number of zeroed attended items is often too large to be helpful as an explanation"

(Serrano and Smith, 2019)

Learning to deceive

- Setup tasks such that it is known, a priori, which tokens are useful for prediction
  - e.g. edit examples of occupation role detection such that "female" tokens would imply a specific label

Attention	Biography	Label
Original	Ms. X practices medicine in Memphis, TN and ... Ms. X speaks English and Spanish.	Physician
Ours	Ms. X practices medicine in Memphis , TN and ... Ms. X speaks English and Spanish.	Physician

(Pruthi et al., 2020)

Learning to deceive

- Train by trying to neglect impermissible tokens $\mathbf{m}$
  - $\mathbf{m}_i = 1$ if x_i is impermissible and 0 otherwise

$$\mathcal{L}(\theta) = NLL(\hat{y}, y) - \lambda \log(1 - \alpha^\top \mathbf{m})$$

(Pruthi et al., 2020)

Learning to deceive

- Train by trying to neglect impermissible tokens $\mathbf{m}$
  - $\mathbf{m}_i = 1$ if x_i is impermissible and 0 otherwise

$$\mathcal{L}(\theta) = NLL(\hat{y}, y) - \lambda \log(1 - \alpha^\top \mathbf{m})$$

Overall, attention can be manipulated with a negligible drop of performance

(Pruthi et al., 2020)

Learning to deceive

- Train by trying to neglect impermissible tokens $\mathbf{m}$
  - $\mathbf{m}_i = 1$ if x_i is impermissible and 0 otherwise

$$\mathcal{L}(\theta) = NLL(\hat{y}, y) - \lambda \log(1 - \alpha^\top \mathbf{m})$$

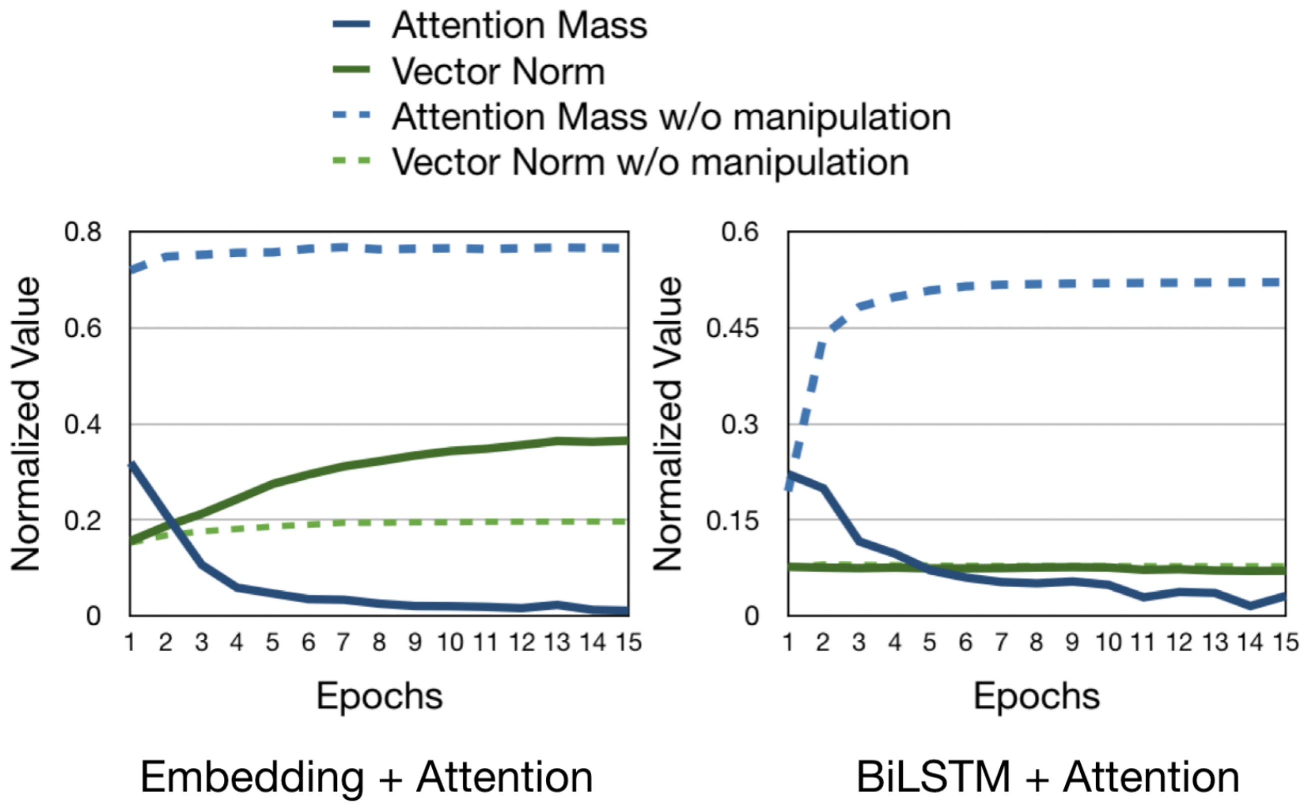
Overall, attention can be manipulated with a negligible drop of performance

- Models find interesting alternative workarounds!

1. RNN-based leak information via recurrent connections
2. Embed-based leak information via vector norms

(Pruthi et al., 2020)

Learning to deceive



1. RNN-based leak information via recurrent connections
2. Embed-based leak information via vector norms

(Pruthi et al., 2020)

Attention is not not explanation

- Questions the conclusions of the previous papers and proposes various explainability tests

- Incomplete adversarial attention experiment

"Jain and Wallace provide alternative distributions which may result in similar predictions, but [...] (ignore the) fact that the model was trained to attend to the tokens it chose"

- Plausible vs faithful explanation

"we hold that attention scores are used as providing an explanation; not the explanation."

(Wiegreffe and Pinter, 2019)

Attention is not not explanation

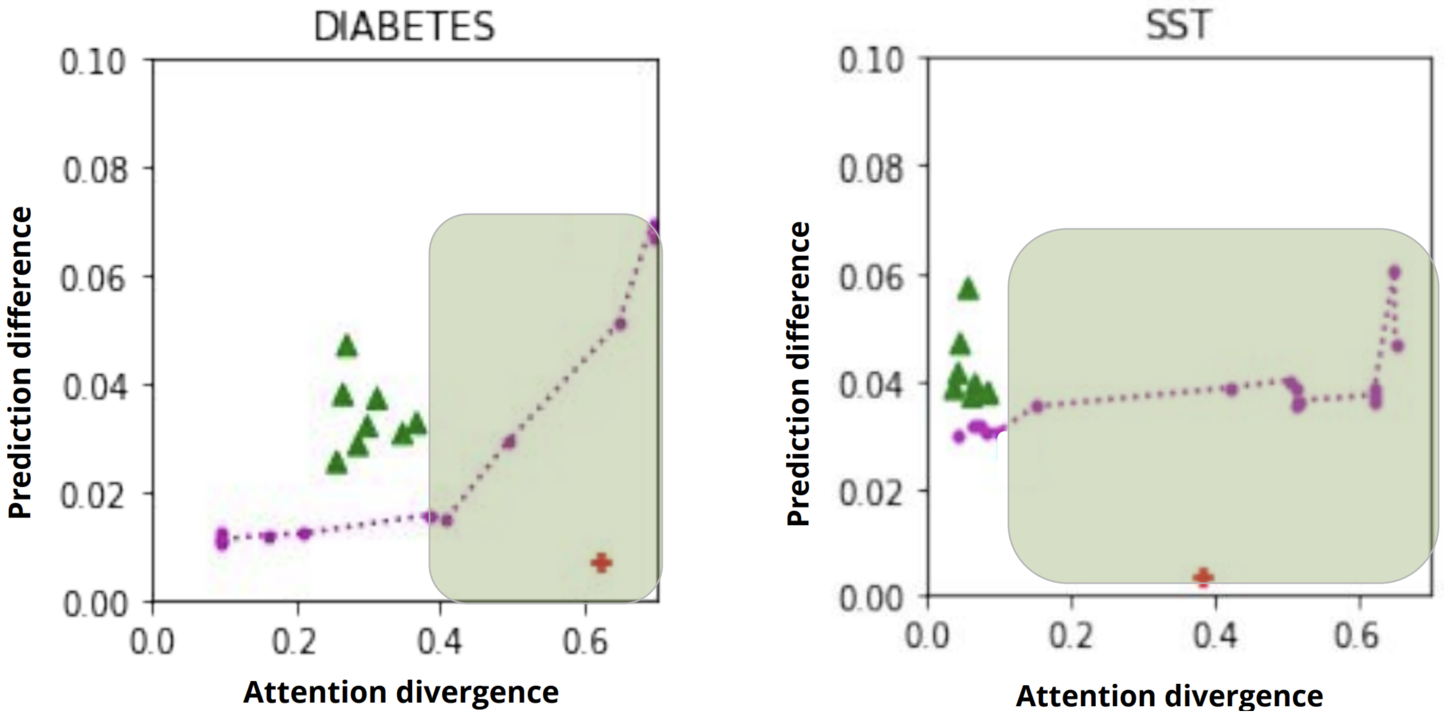
"Train an adversary that **minimizes change in prediction scores**, while **maximizing changes in the learned attention distributions** "

$$\mathcal{L}(\theta) = |\hat{y} - \tilde{y}| - \lambda \underbrace{KL(\alpha || \tilde{\alpha})}_{\text{divergence}}$$

(Wiegreffe and Pinter, 2019)

Attention is not not explanation

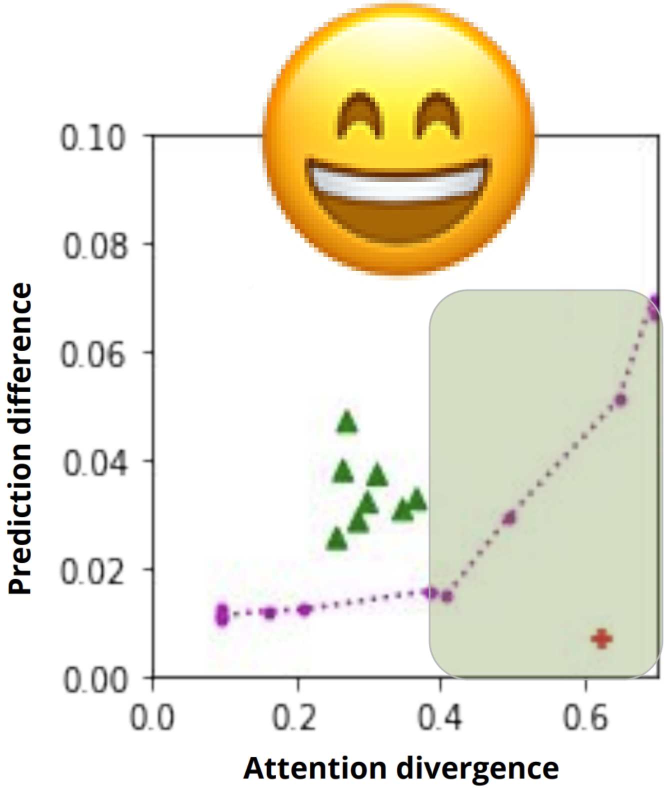
"Train an adversary that **minimizes change in prediction scores**, while **maximizing changes in the learned attention distributions** "

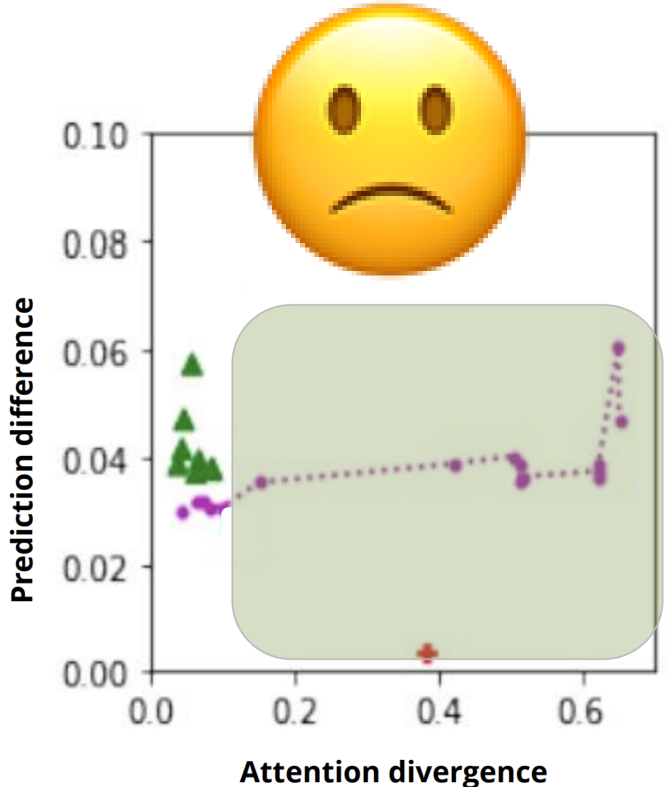


(Wiegreffe and Pinter, 2019)

Attention is not not explanation

"Train an adversary that **minimizes change in prediction scores**, while **maximizing changes in the learned attention distributions** "





(Wiegreffe and Pinter, 2019)

Attention is not not explanation

- Plausible vs faithful explanation (Jacovi and Goldberg, 2020)
  - **Plausibility:** how convincing the explanation is to humans 🤓
  - **Faithfulness:** how accurately it reflects the true reasoning process of the model 🤖

- For attention to be faithful, it should: (Wiegreffe and Pinter, 2019)
  - Be necessary
  - Hard to manipulate
  - Work out of contextualized setting

- Attention is not causation: (Grimsley et al., 2020)
  - "attention is not explanation by **definition**, if a causal explanation is assumed" <===> faithfulness

Towards faithful models?

- Graded notion of faithfulness (Jacovi and Goldberg, 2020)
  - An entire faithful explanation might be impossible
  - Instead, consider the scale of faithfulness

Towards faithful models?

- Graded notion of faithfulness (Jacovi and Goldberg, 2020)
  - An entire faithful explanation might be impossible
  - Instead, consider the scale of faithfulness

- Rudin (2018) defines explainability as a plausible (but not necessarily faithful) reconstruction of the decision-making process

- Riedl (2019) argues that explainability mimics what humans do when rationalizing past actions

Plausibility is also important 🤓

- "Do you believe that highlighted tokens capture the model's prediction?" (Pruthi et al., 2020)
  - Manipulated attentions received a much lower rating than non-manipulated ones

- Attention from BiRNN are very similar to human's attentions (for all evaluated metrics) (Sen et al., 2020)
  - But as *length* increases, they become less similar

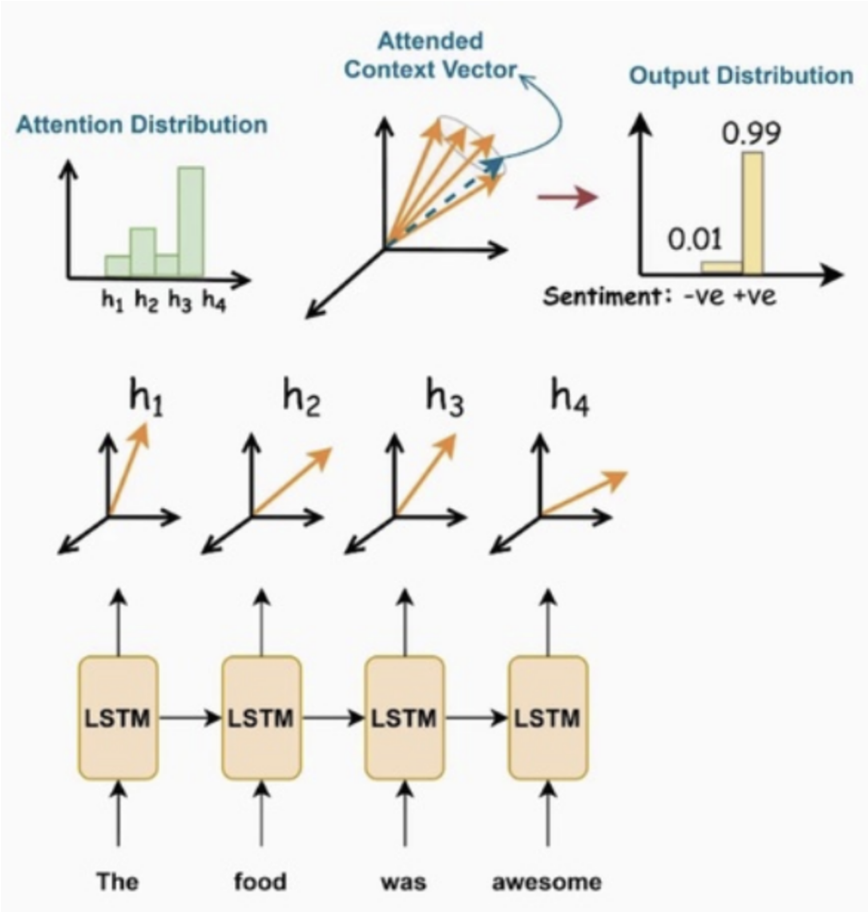
- For text classification, humans find attention (Treviso and Martins, 2020) explanations informative enough to correct predictions
  - But not for natural language inference

Perhaps, we can ask more

- Should attention weights correlate with erasure and gradient measures?
  - Can we regard them as groundtruth for explainability?
  - Are they reliable? (Kindermans et al., 2017)
- Are we evaluating on the right task?
  - Attention is a key piece in tasks like MT and ASR!
- Are we analyzing the right models?
  - What if we limit/increase the contextualization?
  - What if we have latent variables?
  - What are the mechanisms that affect interpretability?

Circumventing attention issues

- Contextualized hidden vectors are very similar

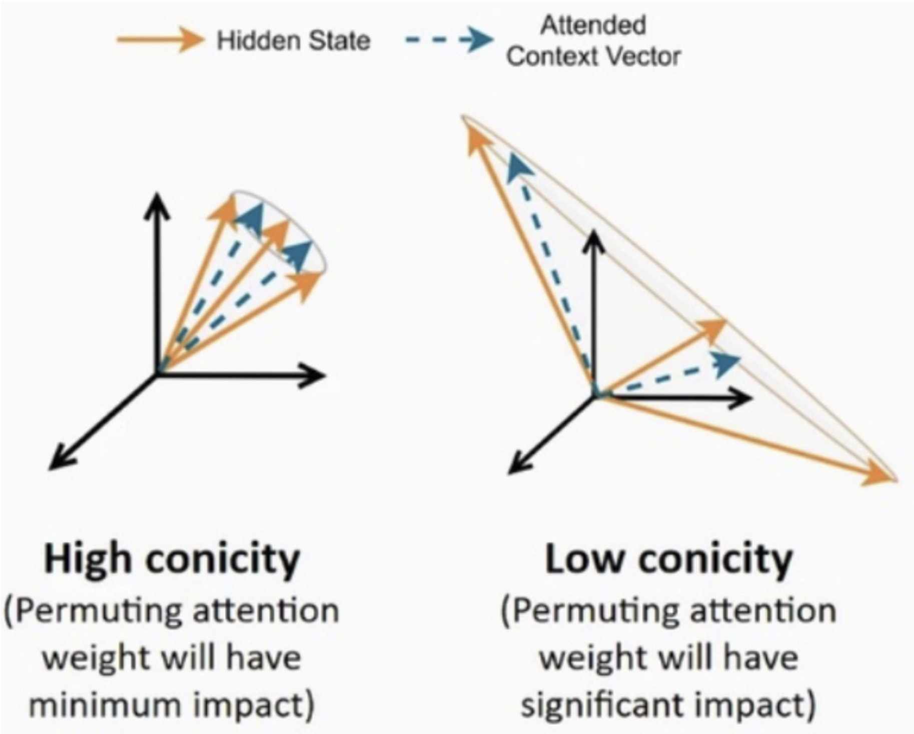


Mohankumar et al., 2020)

Circumventing attention issues

- Contextualized hidden vectors are very similar

$$\text{conicity}(\mathbf{H}) = \frac{1}{m}\sum_{i=1}^{m} \cos(\mathbf{h}_i, \text{mean}(\mathbf{H}))$$



Mohankumar et al., 2020)

Circumventing attention issues

- **Horizontal issue:** contextual vectors leak information
- **Vertical issue:** hidden states lose information about the original input $h_i \iff w_i$

$$\mathcal{L}(\theta) = NLL(\hat{y}, y) - \frac{\lambda}{T} \sum_t \underbrace{||h_t}_{\text{hidden state}} - \underbrace{e_t||_2^2}_{\text{word embedding}}$$

$$\mathcal{L}_{MLM}(\theta) = NLL(\hat{y}, y) + NLL(\underbrace{\hat{w}_{mask}}_{\text{predicted words}}, \underbrace{w_{mask}}_{\text{masked words}})$$

(Tutek and Šnajder, 2020)

Circumventing attention issues

- Faithfulness by construction: rationalizers

generator (ϕ)

$$Z_i \mid \mathbf{x} \sim \mathrm{Bernoulli}(g_{\phi,i}(\mathbf{x}))$$

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x} \odot \mathbf{z})$$

predictor (θ) masked selection!

- Encourage compact and contiguous explanations

$$\Omega(\mathbf{z}) = \underbrace{\lambda_1 \sum_i |z_i|}_{\text{sparsity}} + \underbrace{\lambda_2 \sum_i |z_i - z_{i+1}|}_{\text{contiguity}}$$

(Lei et al., 2016)

Circumventing attention issues

- Faithfulness by construction: rationalizers

$$\min_{\theta,\phi} \; \underbrace{-\mathbb{E}_{P(\mathbf{z}|\mathbf{x};\phi)}\left[\log P(\mathbf{y}|\mathbf{x},\mathbf{z};\theta)\right]}_{\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z})} + \Omega(\mathbf{z})$$

lower bound on the log-likelihood

penalties

- Training is done with REINFORCE (Lei et al., 2016)
  - unbiased but high variance estimator
- HardKuma instead of Bernoulli variables (Bastings et al., 2019)
  - reparameterization trick & controlled sparsity
- Or... we can use α-entmax as *z* (Treviso and Martins, 2020)

Interpreting Transformers

- Probing
  - Are linguistic structure encoded in the representations?
  - "Recent" area but growing fast

  (Voita and Titov, 2020)

  (Pimentel et al., 2020)

- Analyzing attention heads

  (Voita et al., 2019)

  (Correia et al., 2019)

- Analyzing attention flow

  (Abnar and Zuidema, 2020)

  (De Cao et al., 2020)

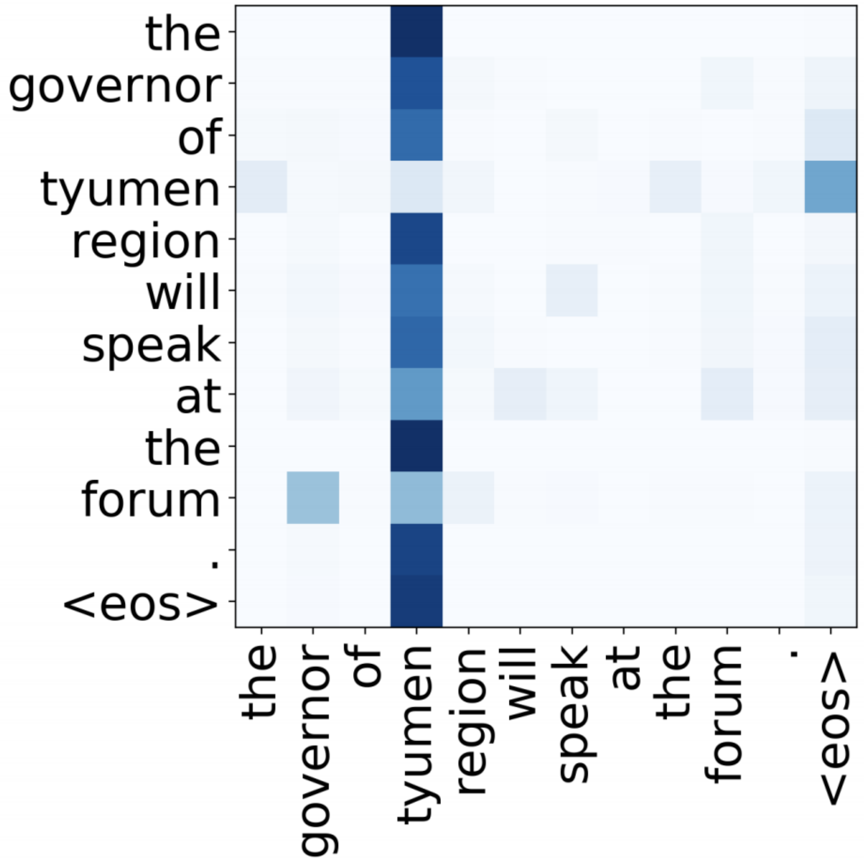
- Analyzing token identifiability across layers

  (Brunner et al., 2020)

  (Kobayashi et al., 2020)

Interpreting Transformers

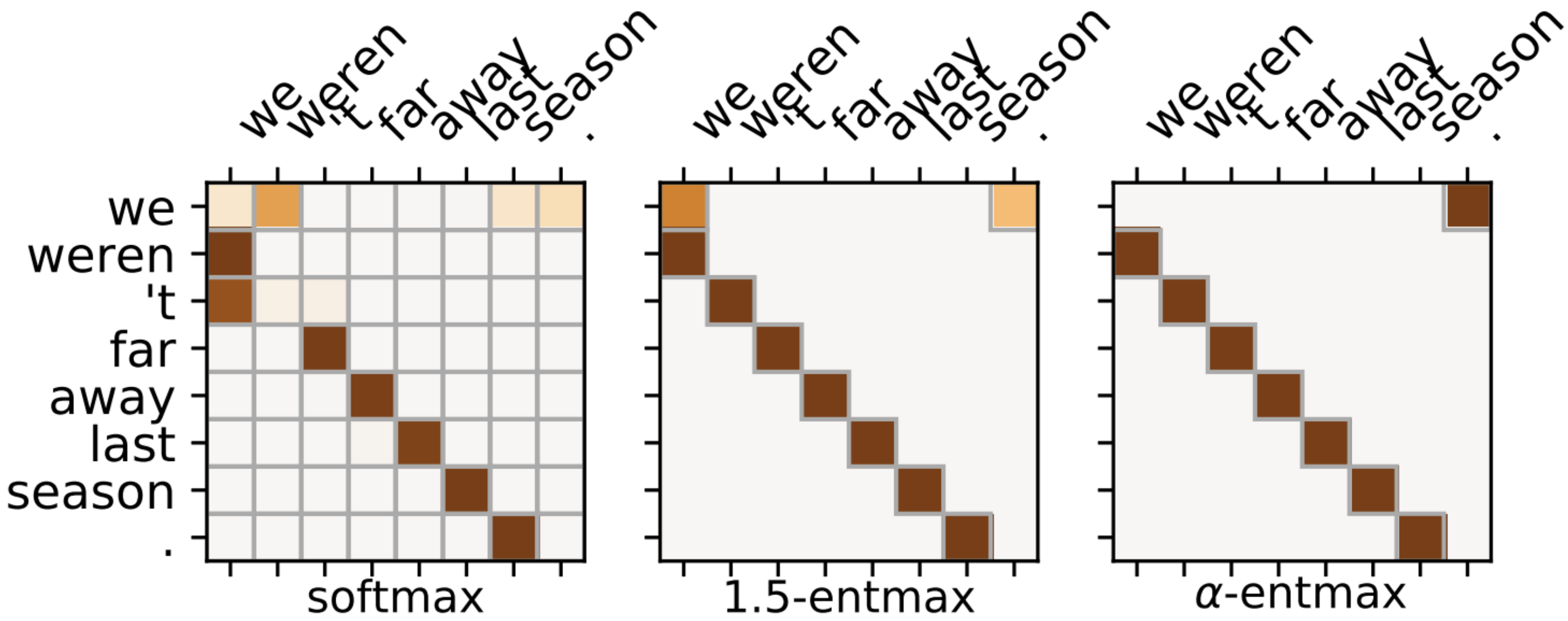
- Specialized head: focus on rare tokens



(Voita et al., 2019)

Interpreting Transformers

- Specialized head: focus on neighbor tokens

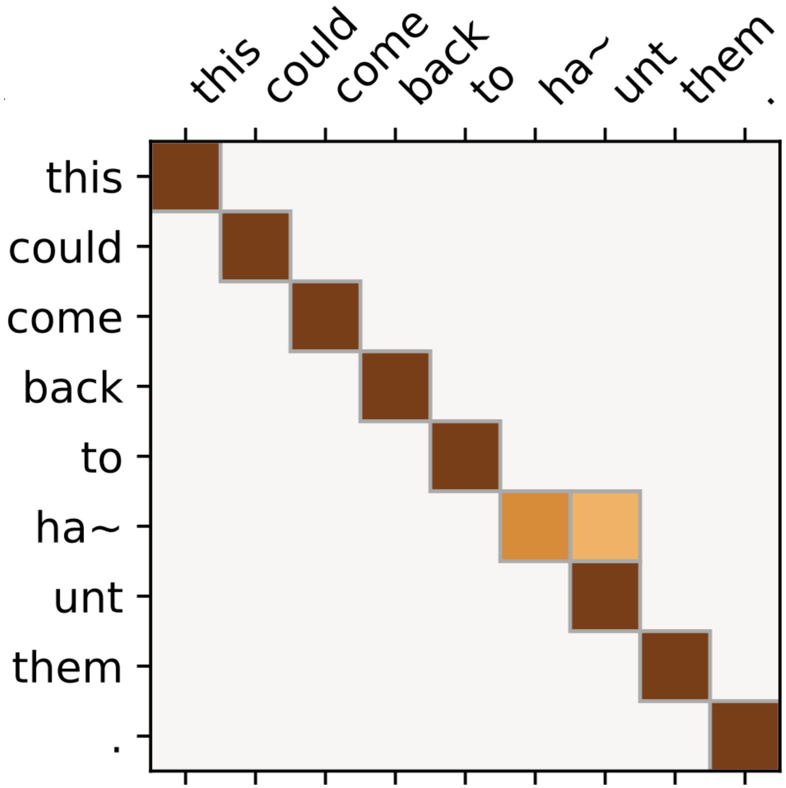


(Voita et al., 2019)

(Correia et al., 2019)

Interpreting Transformers

- Specialized head: merge subword units



(Correia et al., 2019)

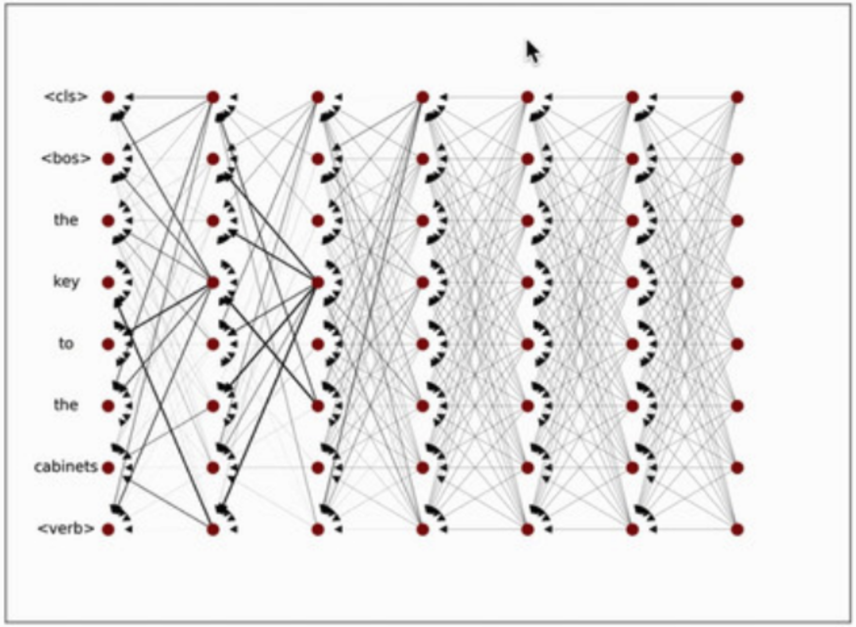
Interpreting Transformers

- Attention flow: consider the Transformer as a DAG structure: attention in $\ell = 1$ is not the same as in $\ell > 1$

- Vertices are tokens

- Edges are connections between $\mathbf{q}_i$ and $\mathbf{k}_j$

- Weights are the attention weights α

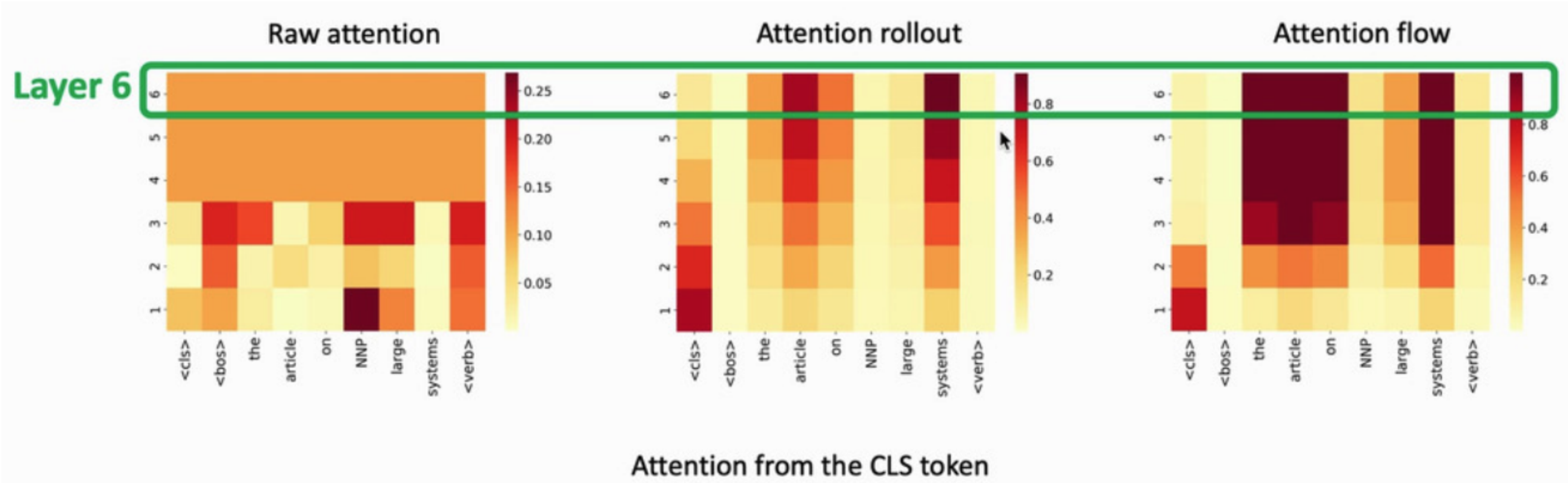


Raw Attention Weights of a 6 layer Transformer trained for sentence classification on subject-verb agreement task of Linzen et al. 2016.

(Abnar and Zuidema, 2020)

Interpreting Transformers

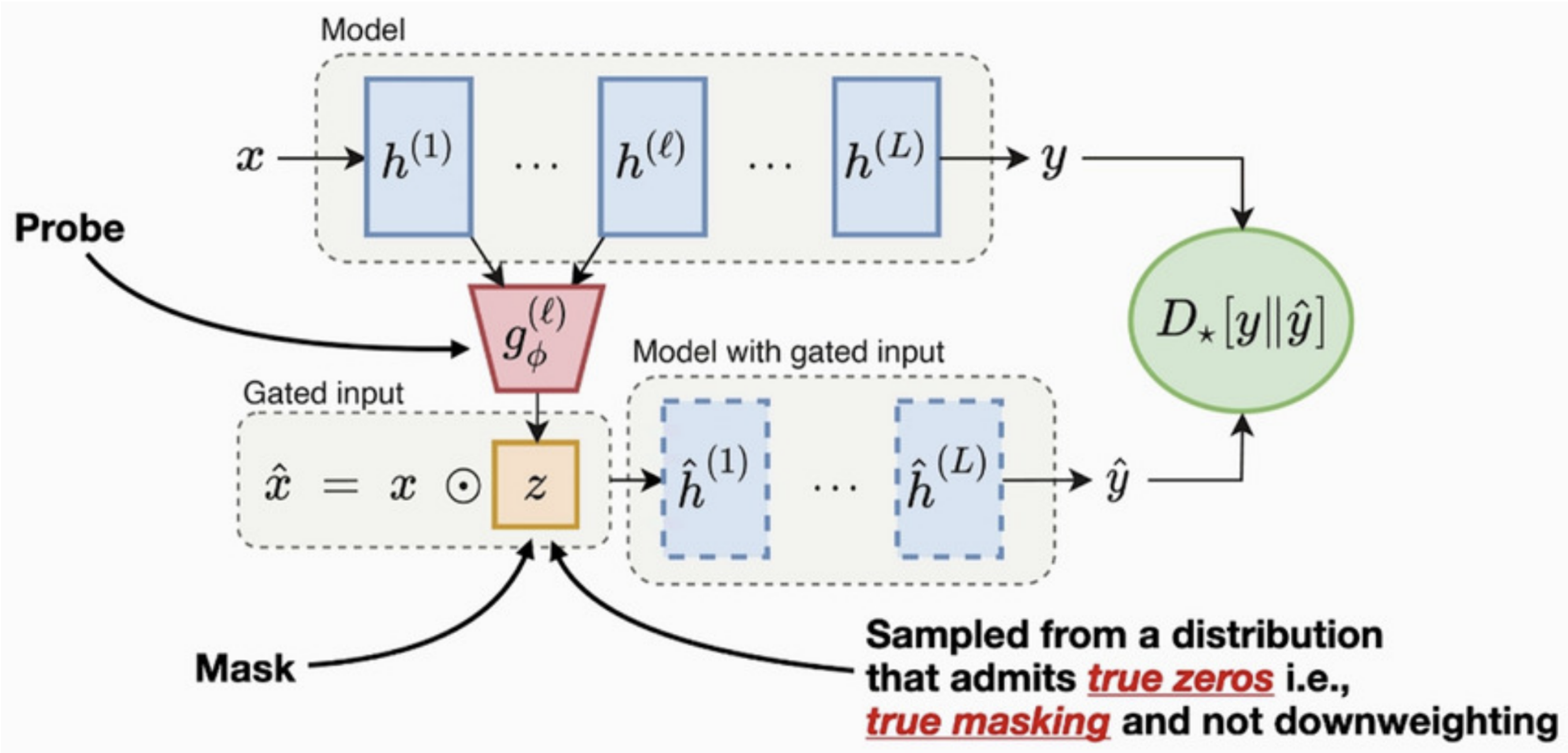
- Attention flow: consider the Transformer as a DAG structure: attention in $\ell = 1$ is not the same as in $\ell > 1$



(Abnar and Zuidema, 2020)

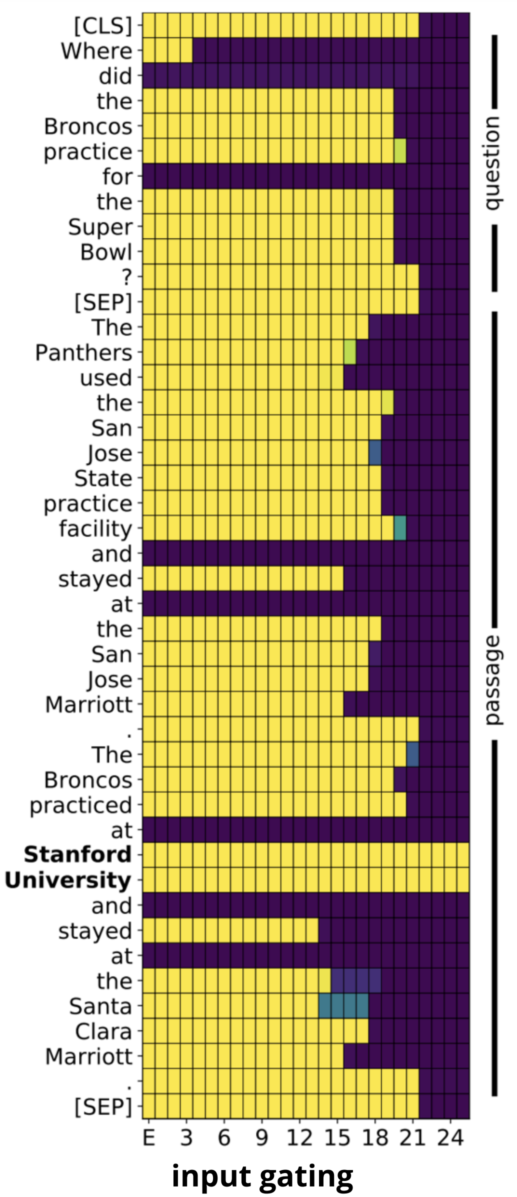
Interpreting Transformers

- Attention flow: which tokens can be ignored as layers go up such that the task performance remains the "same"?



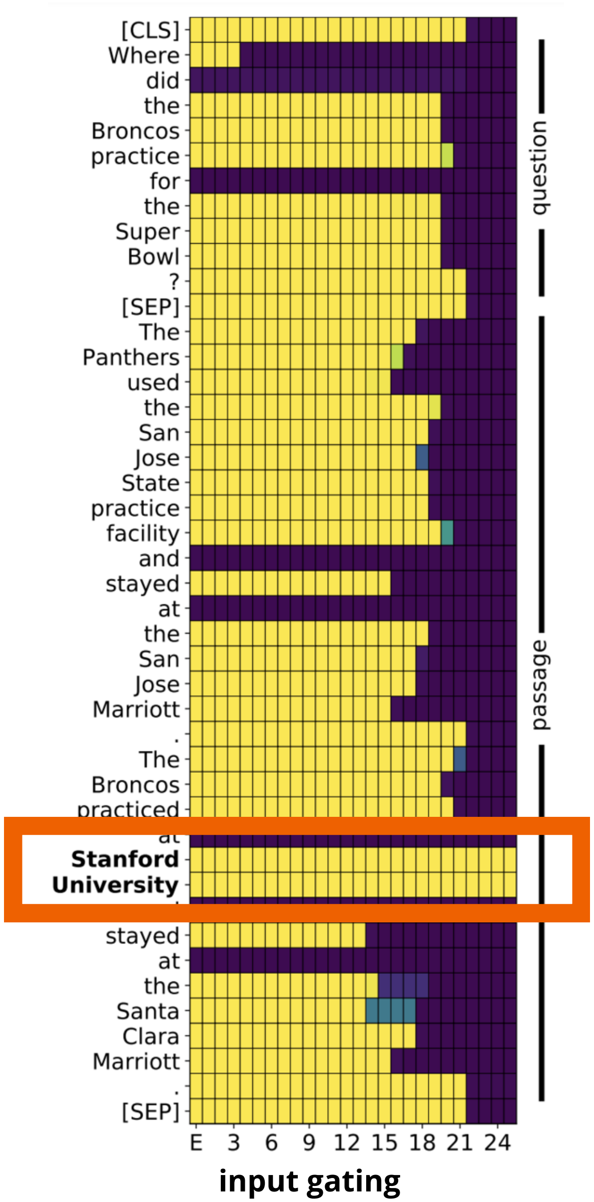
(De Cao et al., 2020)

Interpreting Transformers



(De Cao et al., 2020)

Interpreting Transformers

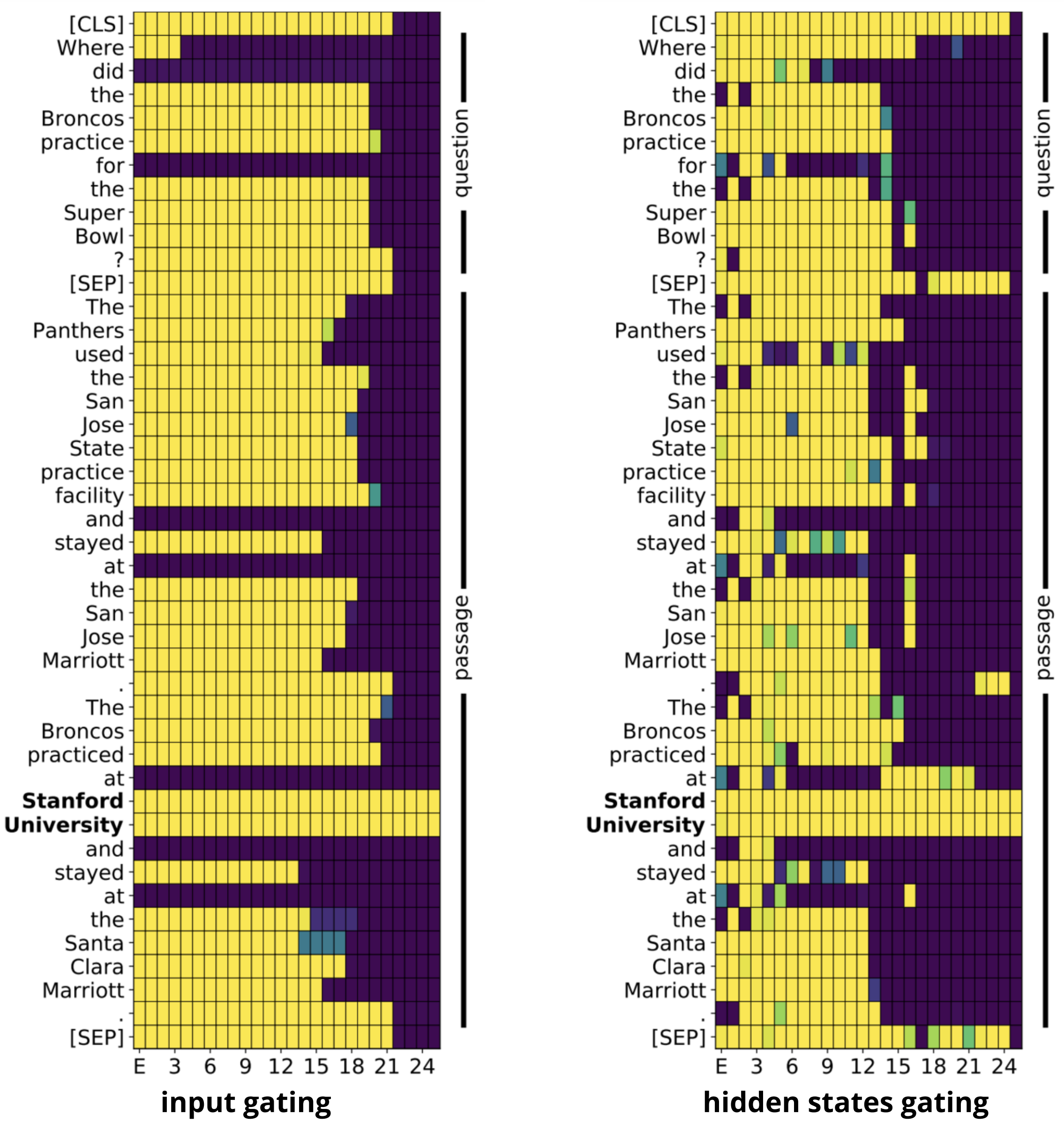


Q: Where did the broncos practice for the Super Bowl?

A: The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott. The Broncos practiced at **Stanford University** and stayed at the Santa Clara Marriott.

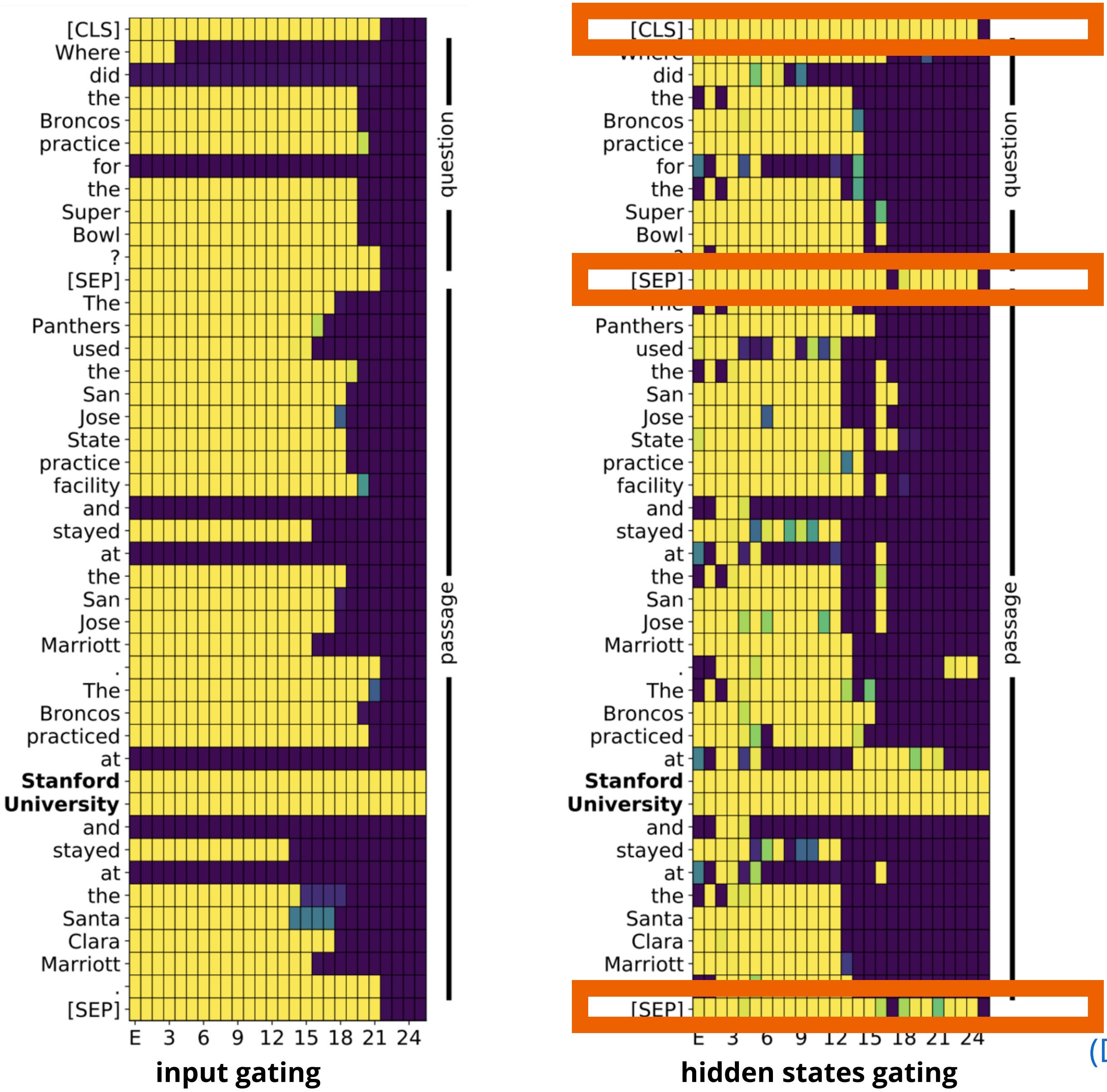
(De Cao et al., 2020)

Interpreting Transformers



(De Cao et al., 2020)

Interpreting Transformers

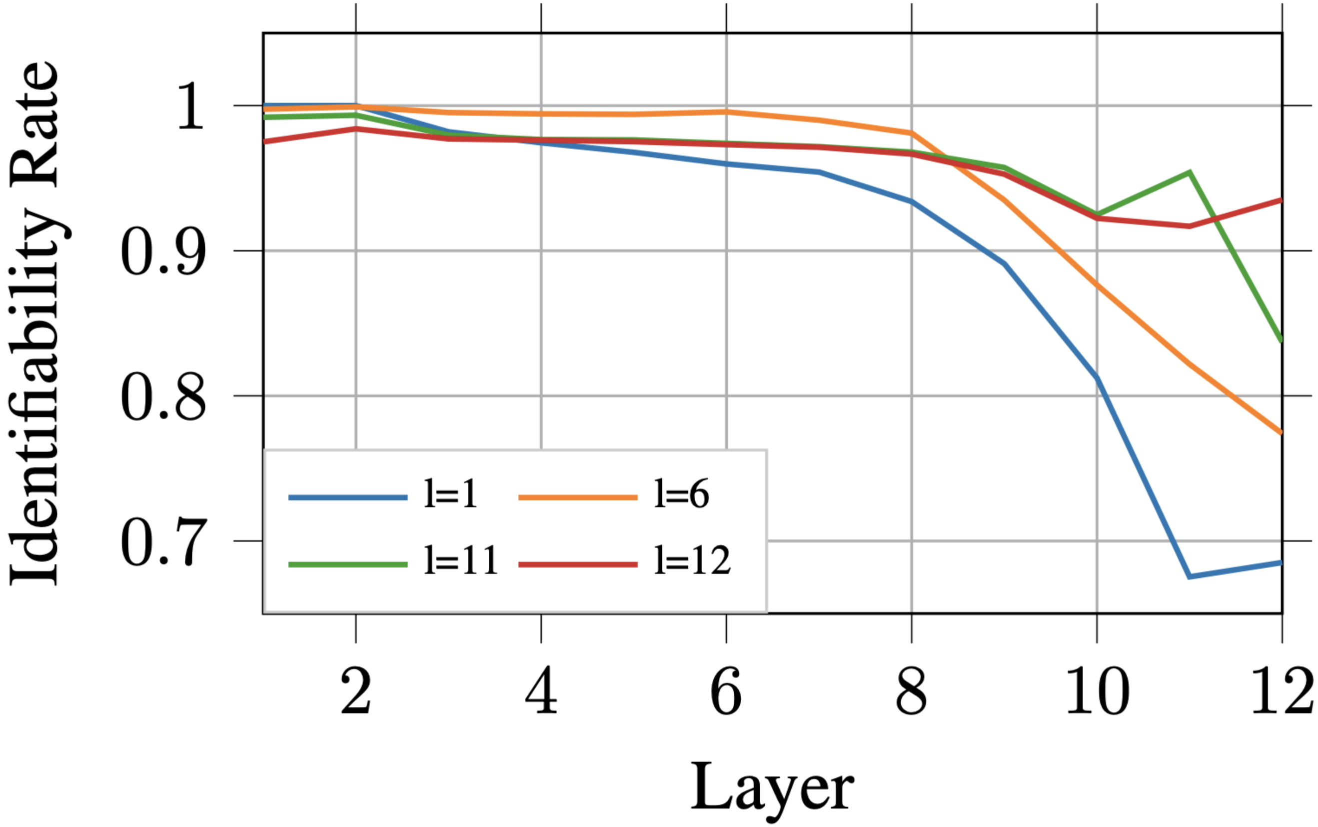


special tokens!

(De Cao et al., 2020)

Interpreting Transformers

- Analyzing token identifiability across layers



(Brunner et al., 2020)

Interpreting Transformers

- Analyzing token identifiability across layers

"when the sequence length is larger than the attention head dimension ($n > d$), self-attention is not unique"

(Brunner et al., 2020)

Interpreting Transformers

- Hot research area!
  - (Vig and Belinkov, 2019)
  - (Tenney et al., 2019)
  - …
- In 2020: Interpretability track for ACL and EMNLP!
- BlackboxNLP workshop: https://blackboxnlp.github.io/
- There are still many contributions to be made!

Conclusions

- Attention is a key ingredient of neural nets
- Attention has many variants with different advantages
- Transformers are "not" just a bunch of self-attention
- Transformers can be improved in terms of speed and memory
  - active research area
- Attention plots can be misleading. Make more analysis!
  - be careful with attention claims
  - active research area
  - open debate situation

Thank you for your attention!