aivancity
SCHOOL FOR
TECHNOLOGY, BUSINESS & SOCIETY
PARIS-CACHAN

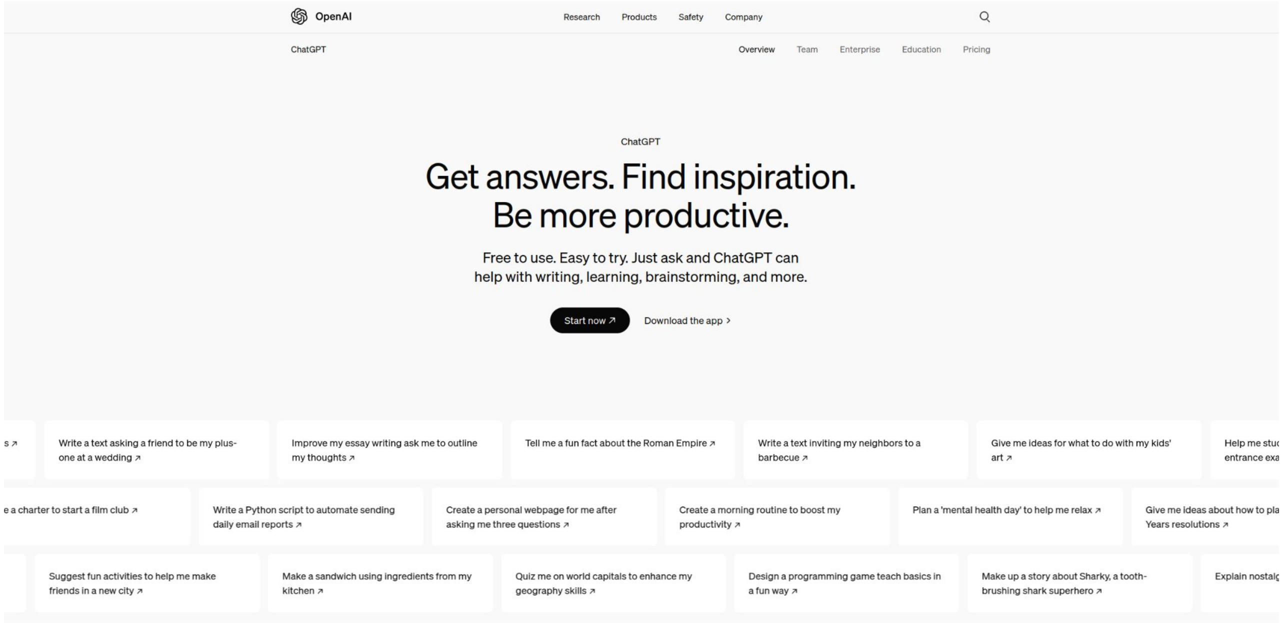
24/10/2024

Natural Language Processing (NLP)

LLM Architectures: Attention Mechanism and Transformers

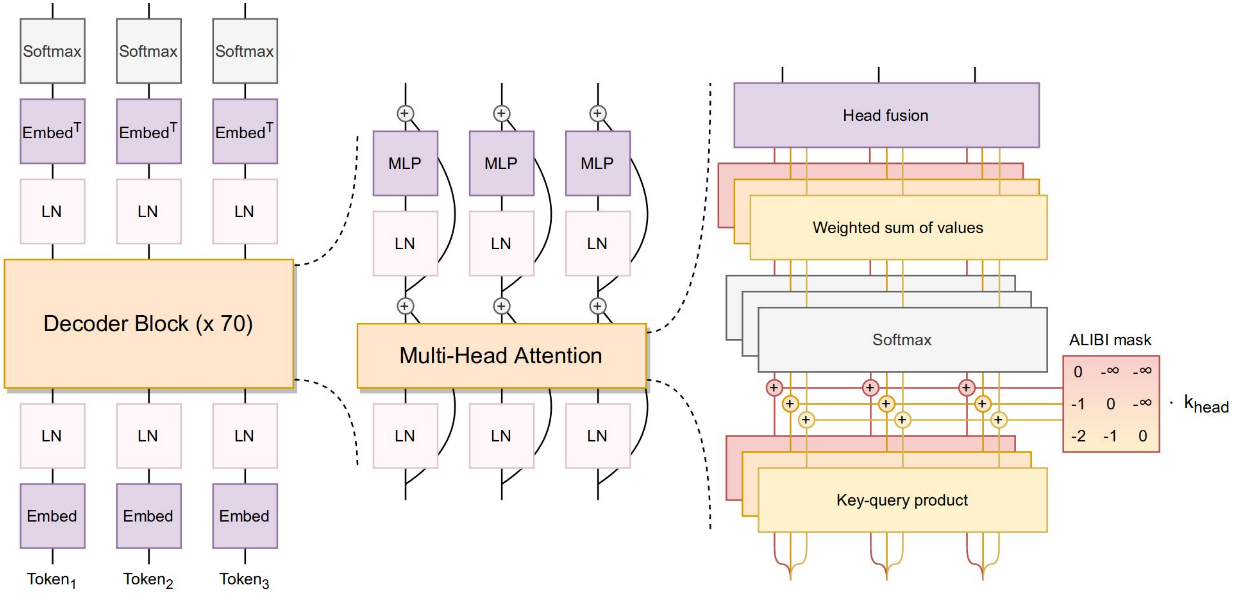
Part 1

Chatbots like ChatGPT rely on LLM



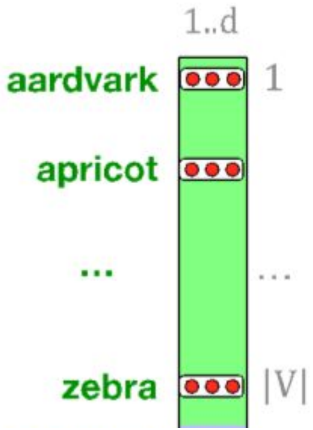
Part 1

LLMs rely on Attention and Transformer

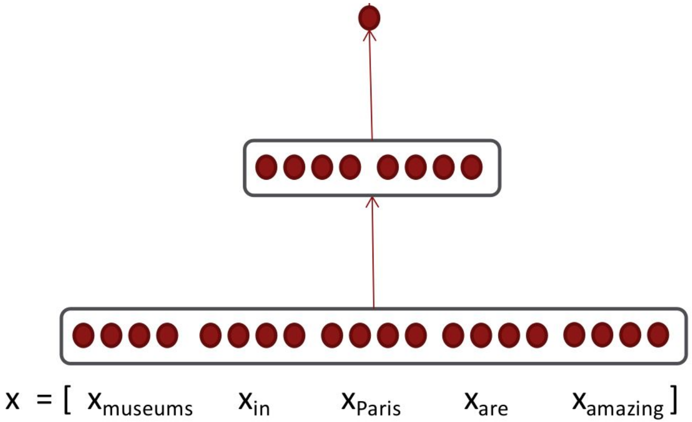


Part 1

What can we do so far?



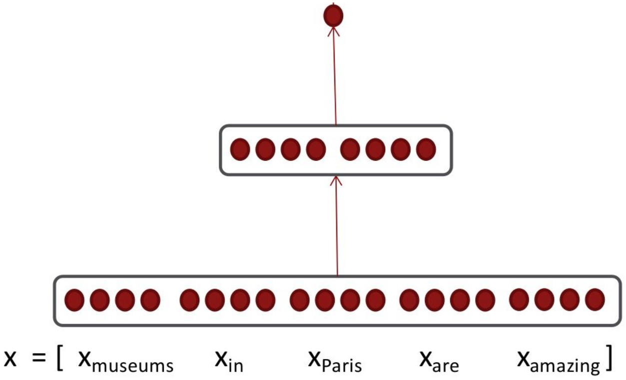
- Embedding matrix: a vector for each word
- Fine for **classification** (e.g. Named Entity Recognition)



Part 1

Continuous bag of words (CBOW)

"bag of words" because does not model word order, puts all words in the same "bag"



x_1: `yes , we have no bananas`

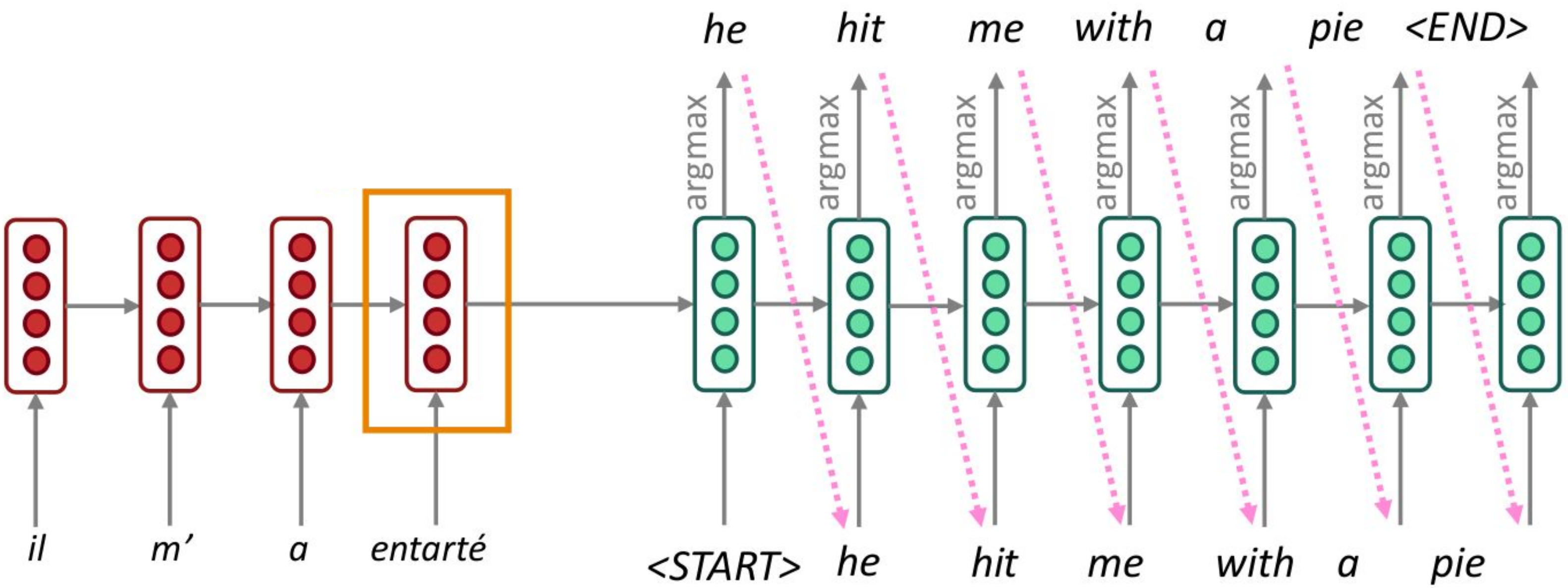
x_2: `say yes for bananas`

x_3: `no bananas , we say`

	1	2	3
,	1	0	1
bananas	1	1	1
for	0	1	0
have	1	0	0
no	1	0	1
say	0	1	1
we	1	0	1
yes	1	1	0

Part 1

Sequence-to-Sequence (Translation)



Part 1

Language Modeling

$p(x \mid \text{START})$		$p(x \mid \text{START I})$		$p(x \mid \cdots \text{went})$		$p(x \mid \cdots \text{to})$		$p(x \mid \cdots \text{the})$		$p(x \mid \cdots \text{park})$		$p(x \mid \text{START I went to the park.})$	
The	3 %	think	11 %	to	35 %	the	29 %	bathroom	3 %	and	14 %	I	21 %
When	2,5 %	was	5 %	back	8 %	a	9 %	doctor	2%%	with	9	It	6
They	2 %	went	2 %	into	5 %	see	5 %	hospital	2 %	,	8 %	The	3 %
…	…	am	1 %	through	4 %	my	3 %	store	1,5 %	to	7 %	There	3 %
I	1 %	will	1 %	out	3 %	bed	2 %	…	…	…	…	…	…
…	…	like	0,5 %	on	2 %	school	1 %	park	0,5 %	.	6 %	STOP	1 %
Banana	0,1 %	…	…	…	…%	…	…	…	…	…	…	…	…

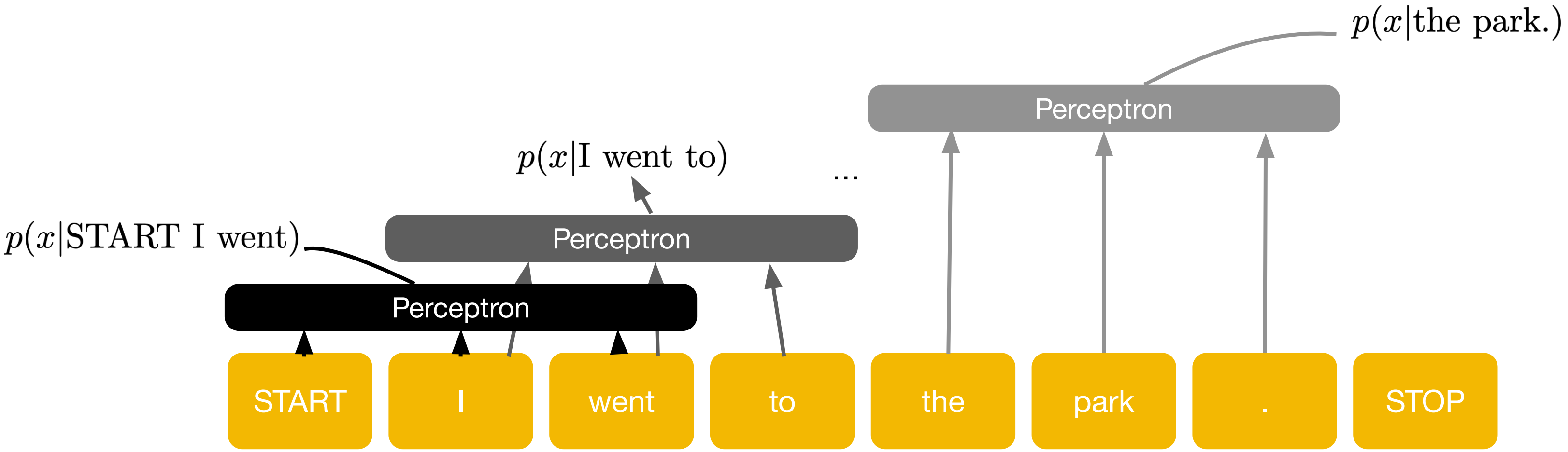
How to model this?

START | I | went | to | the | park | . | STOP

Part 1

v0: sliding window

- Sliding window of size N, like CBOW or n-grams (next class):
  (N − 1)th-order Markov assumption (prediction only depends on N-1 last)

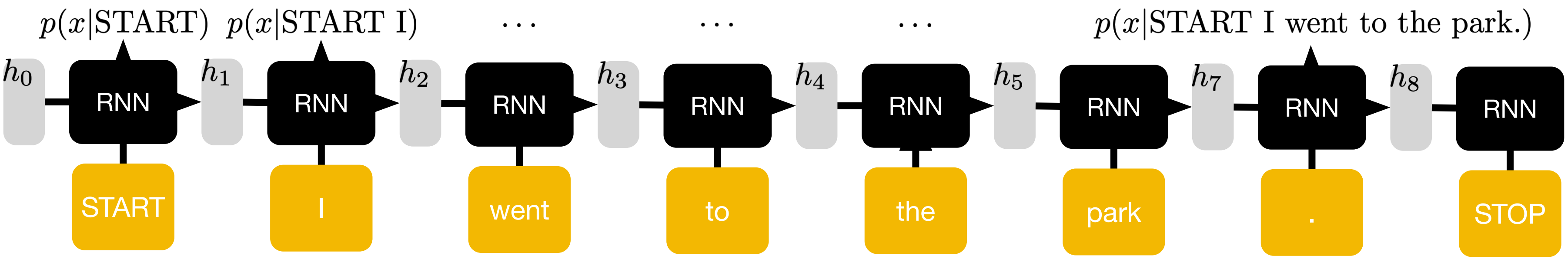


(Bengio et al., 2000)

Part 1

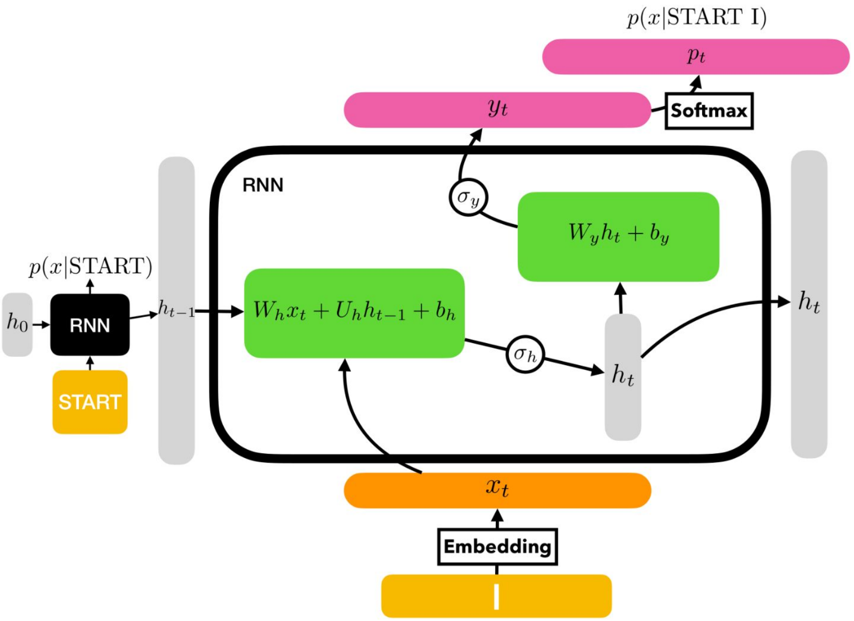
v1: Recurrent Neural Networks (RNN)

- One RNN is applied recursively to the sequence
- Inputs: previous hidden state h_{t-1}, observation x_t
- Outputs: next hidden state h_t, (optionally) output y_t
- Memory about history is passed through hidden states



Part 1

v1: Recurrent Neural Networks (RNN)



Variables:

x_t: input (embedding) vector

y_t: output vector (logits)

p_t: probability over tokens

h_{t-1}: previous hidden vector

h_t: next hidden vector

σ_h: activation function for hidden state

σ_y: output activation function

Equations:

$$h_t := \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t := \sigma_y(W_y h_t + b_y)$$

$$p_{t_i} = \frac{\exp(y_{t_i})}{\sum_{i=j}^{d} \exp(y_{t_j})}$$

Generation as a Sequence of Classifications

$$\hat{y}_1 = \begin{array}{r} p(\text{STOP}|\text{START}) \\ p(\text{The}|\text{START}) \\ p(\text{I}|\text{START}) \\ \\ \\ p(\text{apple}|\text{START}) \end{array} \begin{bmatrix} .01 \\ .03 \\ .1 \\ .001 \\ \vdots \\ .002 \end{bmatrix} \qquad y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

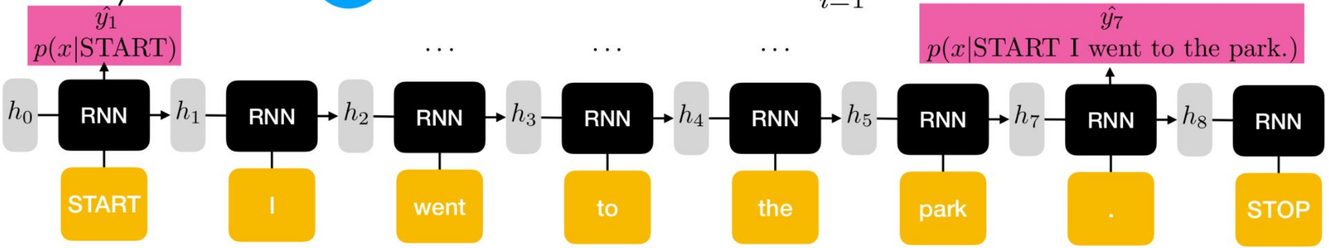
2. Run model on (batch of) x's from data $\mathcal{D}$ to get probability distributions $\hat{y}$

3. Calculate loss compared to true y's (Cross Entropy Loss)

$$\hat{y}_7 = \begin{bmatrix} .2 \\ .03 \\ .12 \\ .01 \\ \vdots \\ .001 \end{bmatrix} \qquad y_7 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

L_{CE}

$$L_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$



Cross-Entropy

$$L_{\mathrm{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

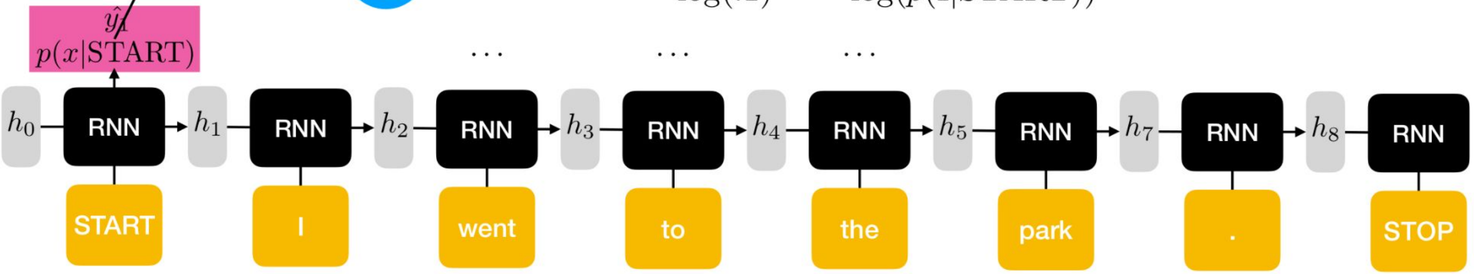
3. Calculate loss compared to true y's (Cross Entropy Loss)

$$\begin{array}{r} p(\mathrm{STOP}|\mathrm{START}) \\ p(\mathrm{The}|\mathrm{START}) \\ p(\mathrm{I}|\mathrm{START}) \\ \hat{y_1} = \\ \\ p(\mathrm{apple}|\mathrm{START}) \end{array} \begin{bmatrix} .01 \\ .03 \\ .1 \\ .001 \\ \vdots \\ .002 \end{bmatrix} \qquad y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

(Actual observed word)

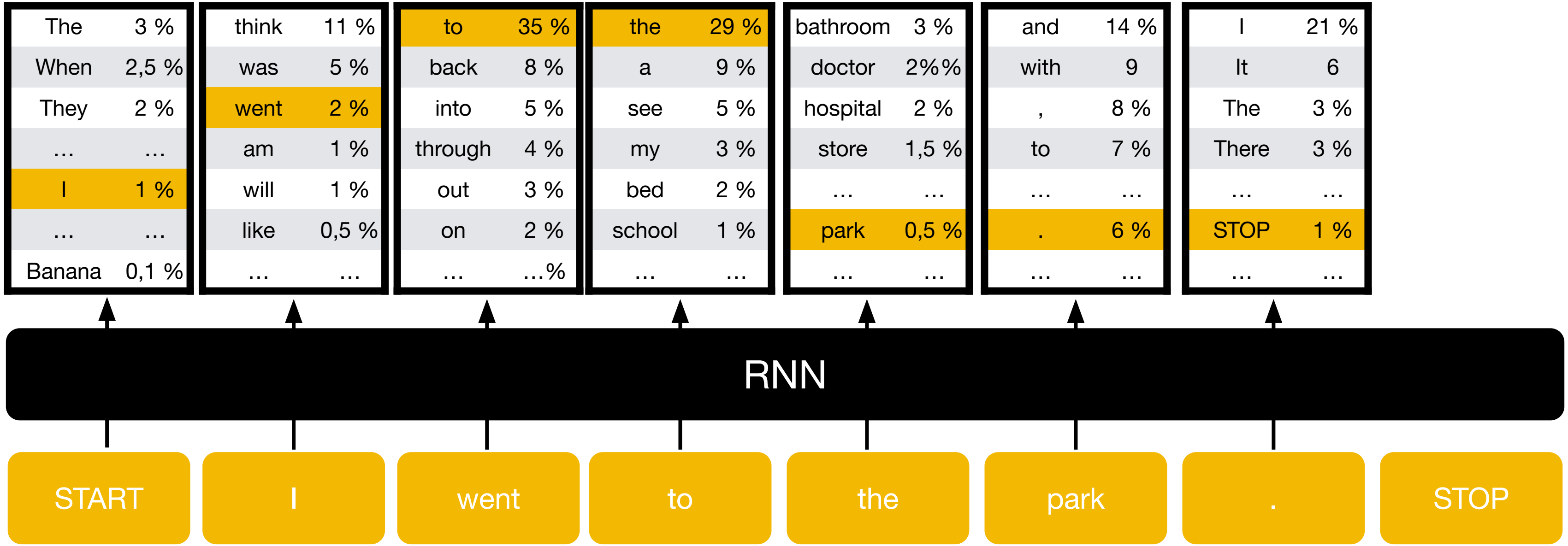
L_{CE}

$$\begin{aligned} L_{\mathrm{CE}}(y_1, \hat{y_1}) &= -0 * \log(.01) - 0 * \log(.03) - 1 * -\log(.1) - \cdots - 0 * \log(.002) \\ &= -\log(.1) = -\log(p(\mathrm{I}|\mathrm{START})) \end{aligned}$$

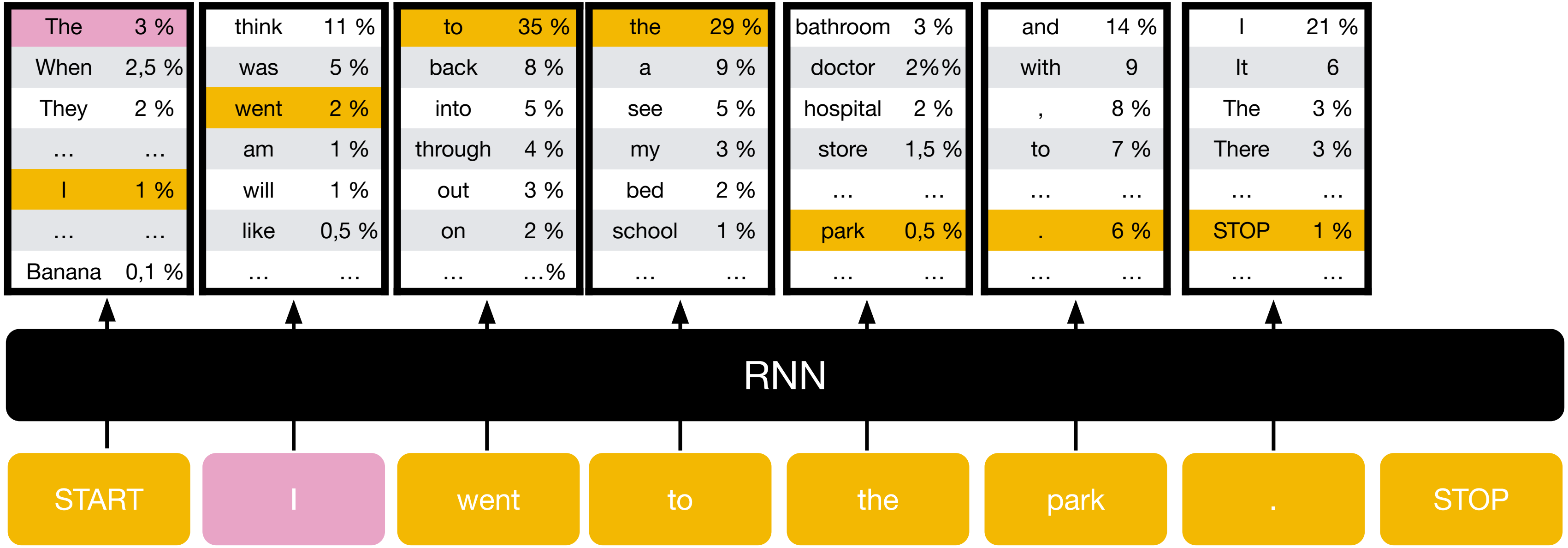


Part 1

Teacher Forcing

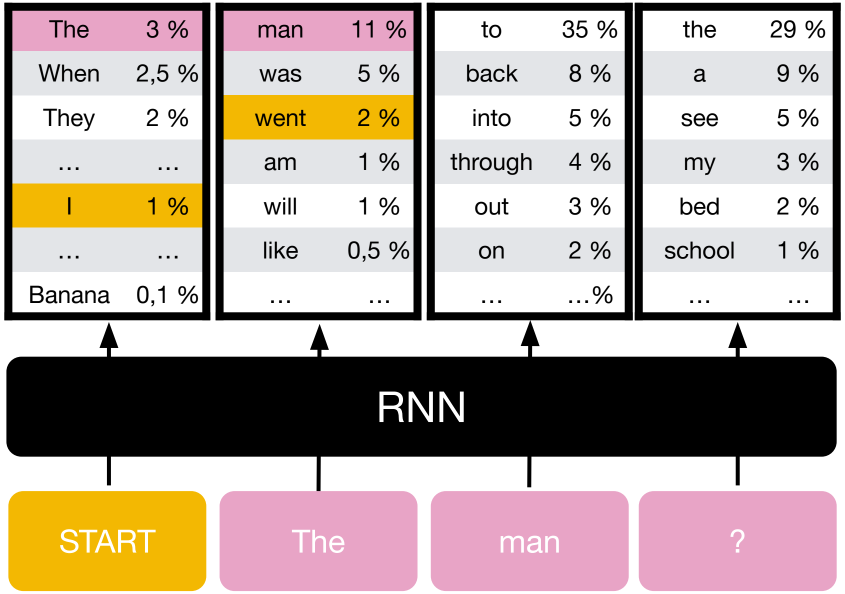


Train-test mismatch: exposure bias



Part 1

Train-test mismatch: exposure bias



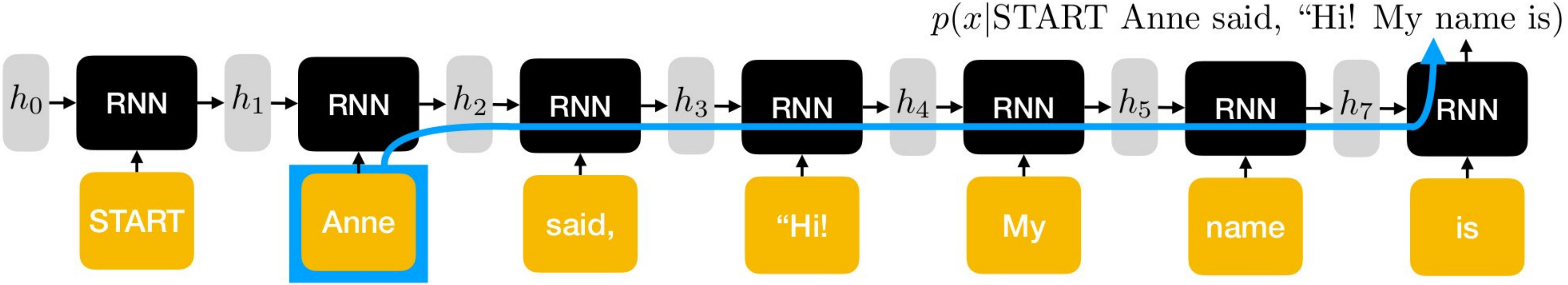
- Keep in mind: with RNN its trivial to train the model with its own generation instead of teacher forcing
- reduces exposure bias

Part 1

RNN limit 1: vanishing gradients

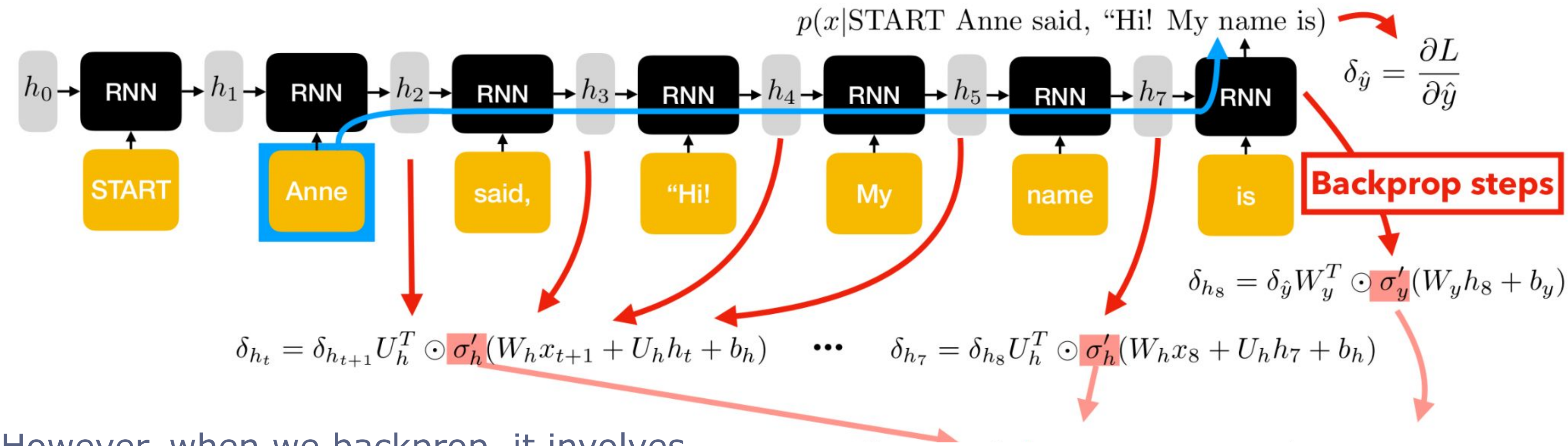
- What word is likely to come next for this sequence?

  Anne said, “Hi! My name is



- Need relevant information to flow across many time steps
- When we backpropagate, we want to allow the relevant information to flow

RNN limit 1: vanishing gradients



However, when we backprop, it involves multiplying a chain of computations from time t_1 to time t_7

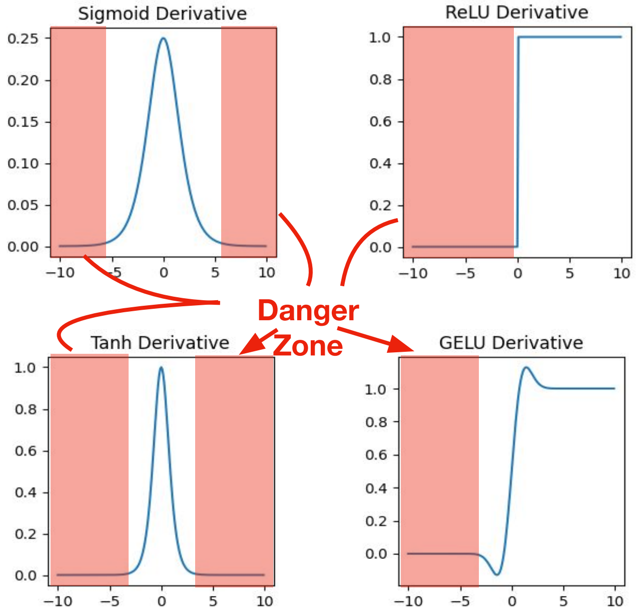
If any of the terms are close to zero, the whole gradient goes to zero (vanishes!)

Part 1

RNN limit 1: vanishing gradients

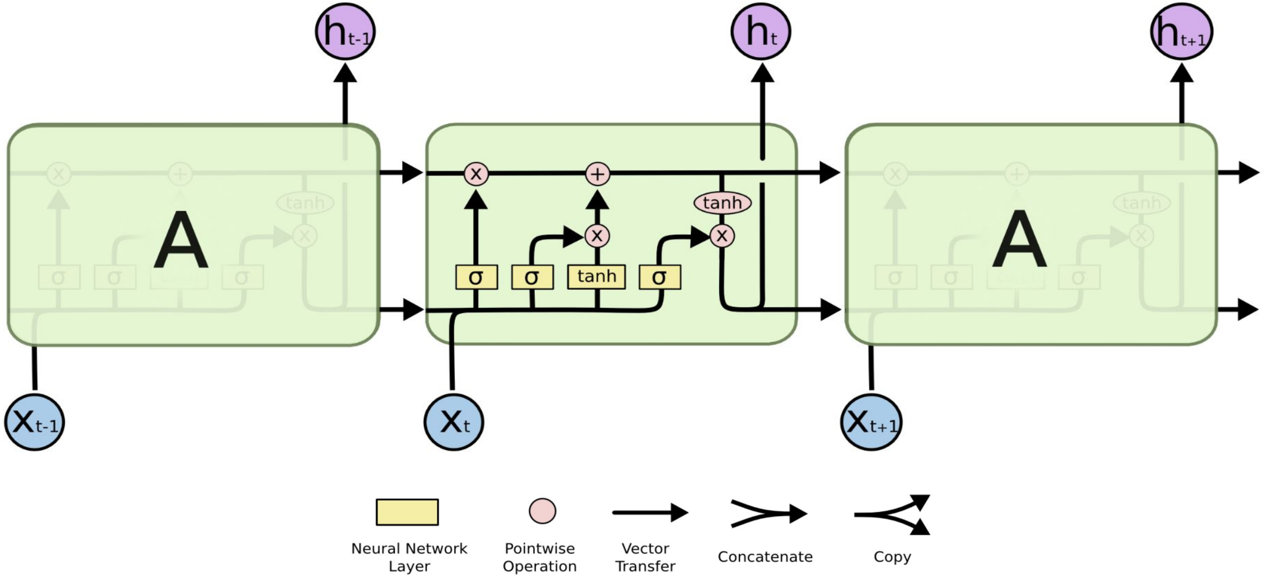
$$\delta_{h_t} = \delta_{h_{t+1}} U_h^T \odot \sigma'_h(W_h x_{t+1} + U_h h_t + b_h)$$

- If any of the terms are close to zero, the whole gradient goes to zero (vanishes!)
- This happens often for many activation functions… the gradient is close to zero when outputs get very large or small
- The more time steps back, the more chances for a vanishing gradient



Part 1

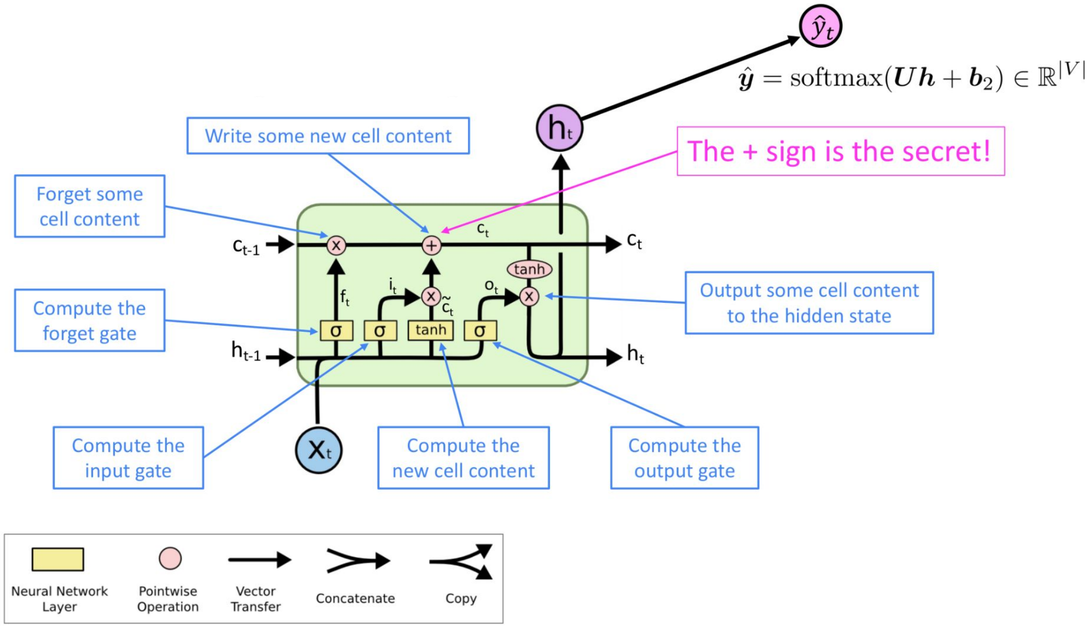
v1.1: Long-Short Term Memory (LSTM)



(Hochreiter & Schmidhuber, 1997)

Part 1

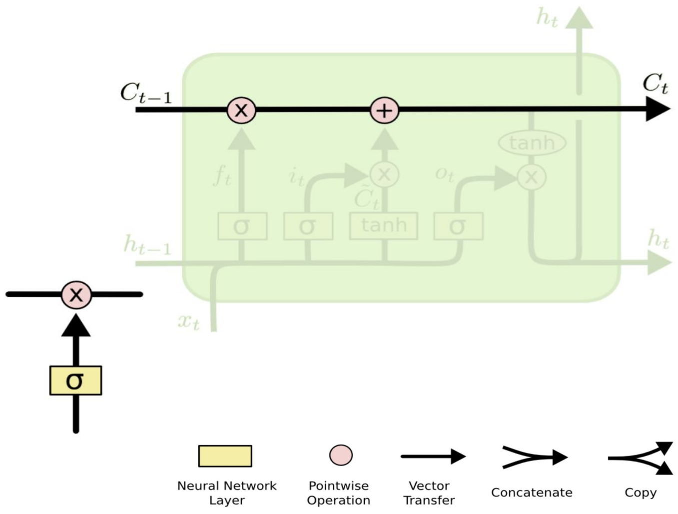
v1.1: Long-Short Term Memory (LSTM)



(Hochreiter & Schmidhuber, 1997)

v1.1: Long-Short Term Memory (LSTM)

Cell state (**long-term memory**): allows information to flow with only small, **linear interactions** (good for gradients!)



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

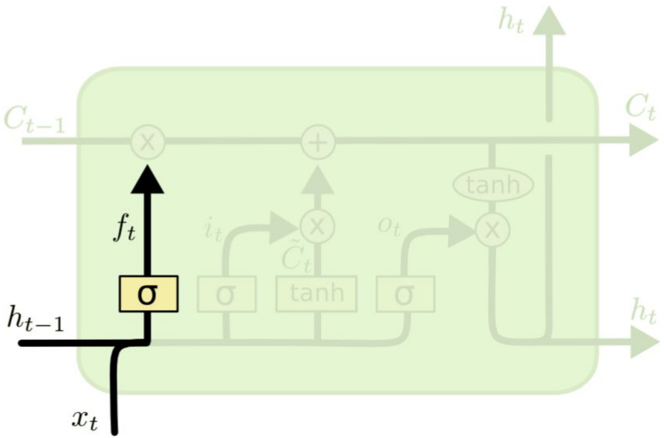
$c_t \in \mathbb{R}^h$: cell state vector

(Hochreiter & Schmidhuber, 1997)

Part 1

v1.1: Long-Short Term Memory (LSTM)

Input Gate Layer: Decide what information to "**forget**"



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

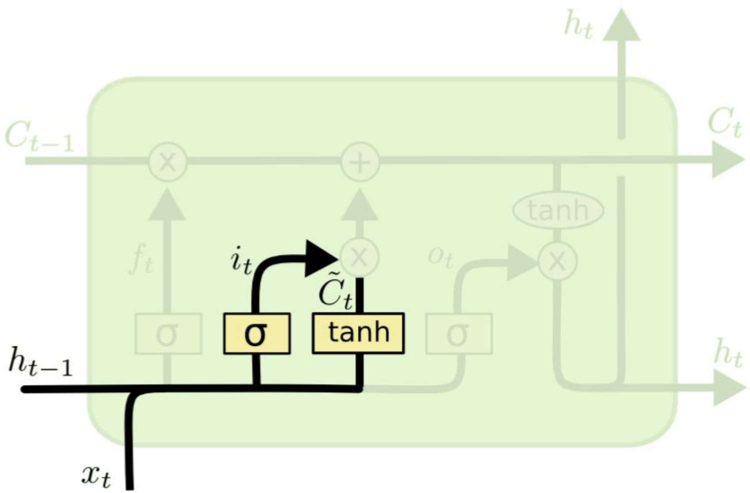
$c_t \in \mathbb{R}^h$: cell state vector

(Gers et al., 2000)

Part 1

v1.1: Long-Short Term Memory (LSTM)

Candidate state values: Extract candidate information to put into the cell vector: "**remember**"



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

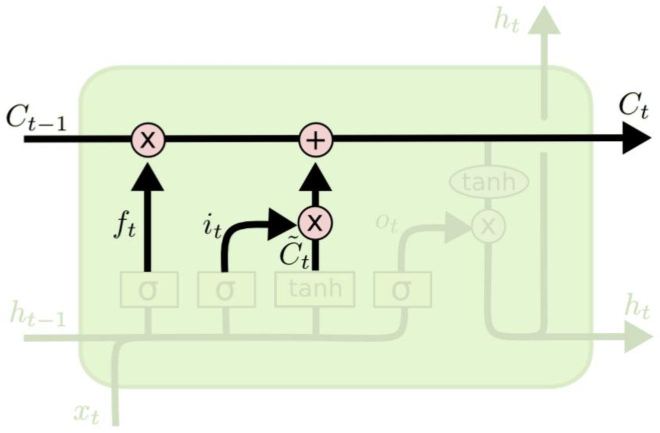
$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

$c_t \in \mathbb{R}^h$: cell state vector

Part 1

v1.1: Long-Short Term Memory (LSTM)

Update cell: "Forget" the information we decided to forget and update with new candidate information



If ***f_t*** is:

- High: we remember more previous info
- Low: we "forget" more info

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

If ***i_t*** is:

- High: we add more new info
- Low: we add less new info

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

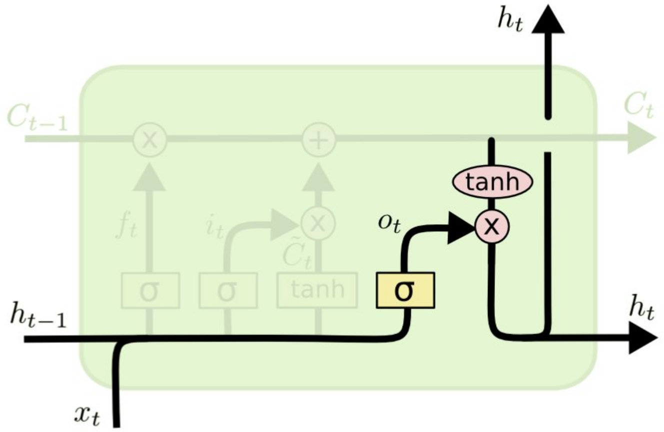
$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

$c_t \in \mathbb{R}^h$: cell state vector

Part 1

v1.1: Long-Short Term Memory (LSTM)

Output/**Short-term Memory** (as in RNN):
Pass information onto the next state/for use in output (e.g., probabilities)



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

$c_t \in \mathbb{R}^h$: cell state vector

Part 1

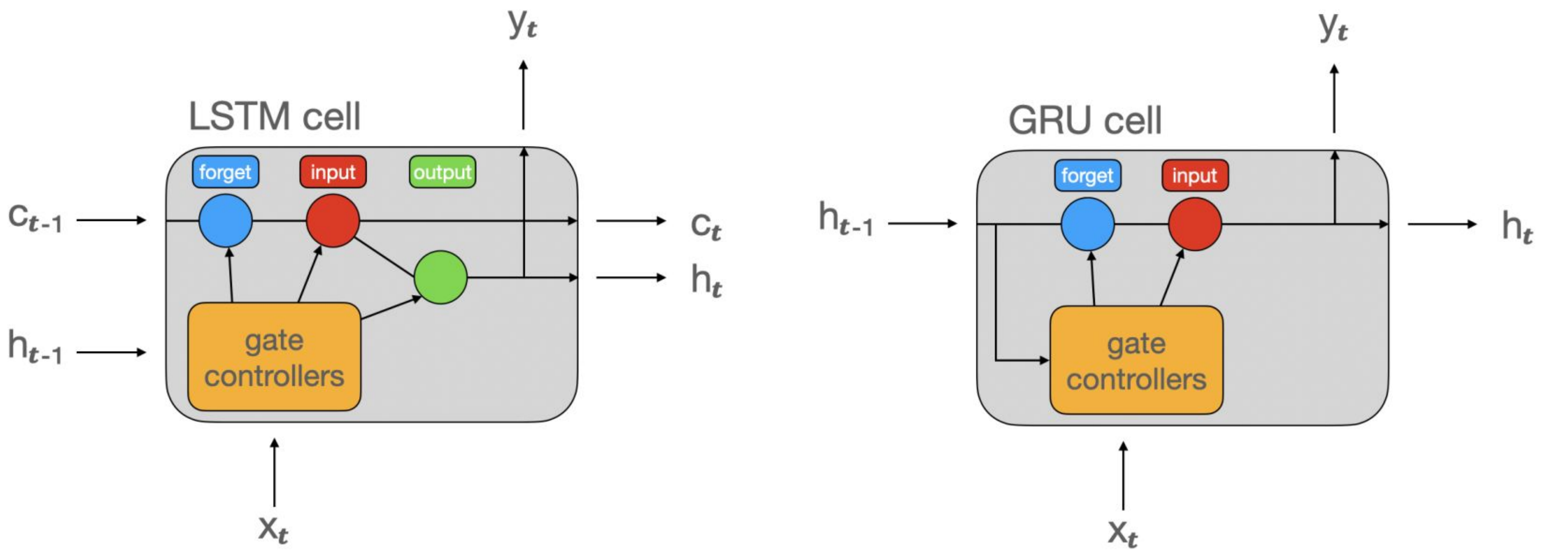
LSTM for Machine Translation

Type	Sentence
Our model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .
Our model	" Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu' ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu' ils pourraient interférer avec les tours de téléphone cellulaire lorsqu' ils sont dans l' air " , dit UNK .
Truth	" Les téléphones portables sont véritablement un problème , non seulement parce qu' ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d' après la FCC , qu' ils pourraient perturber les antennes-relais de téléphonie mobile s' ils sont utilisés à bord " , a déclaré Rosenker .

Sutskever et al. (2014)

Part 1

Gated Recurrent Units (GRU)

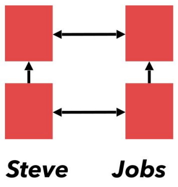


Cho et al. (2014)

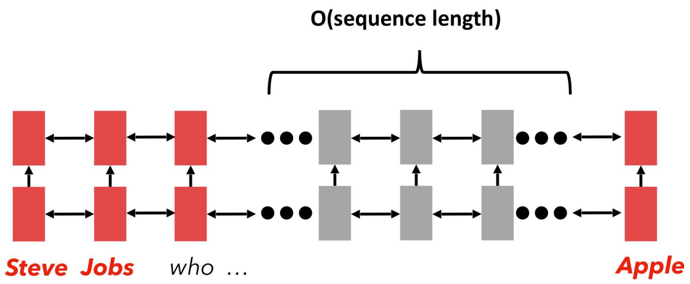
Part 1

RNN limit 2: Linear Interaction Distance

- RNNs are unrolled left-to-right.
- Linear locality is a useful heuristic: nearby words often affect each other's meaning!

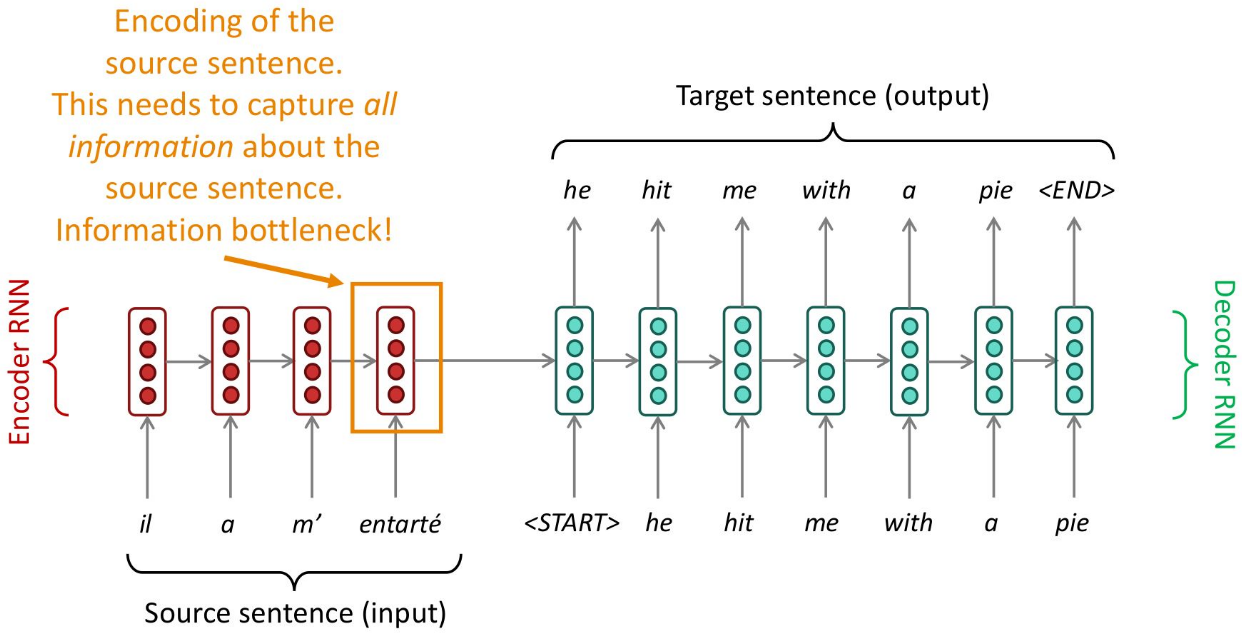


- Better with LSTM then RNN, but still:
  - Failing to capture long-term dependencies
  - Vanishing gradient

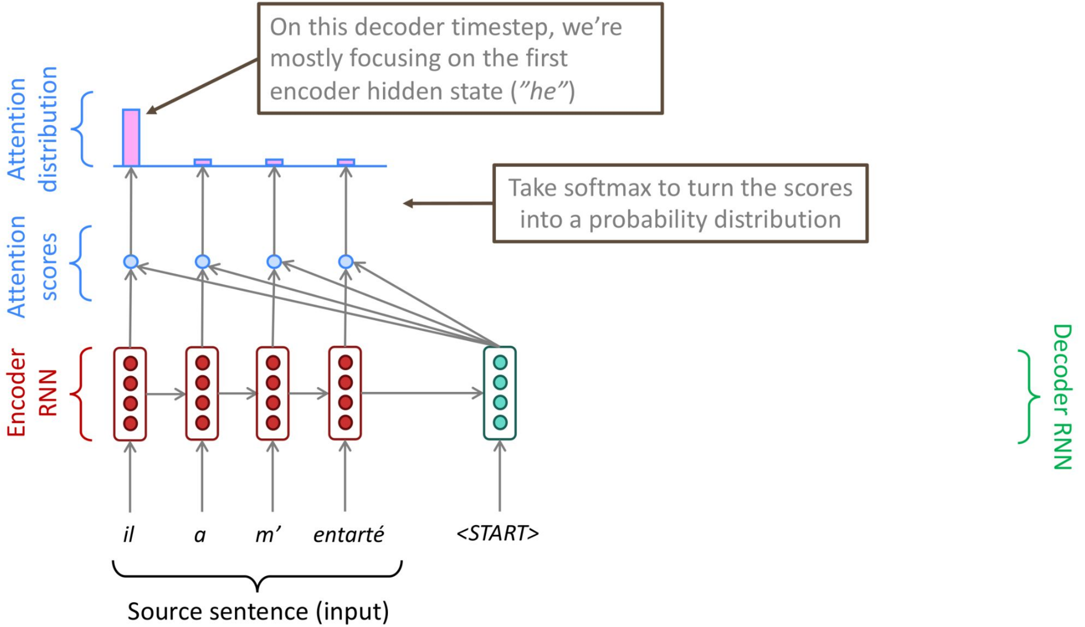


Part 1

Why Attention?



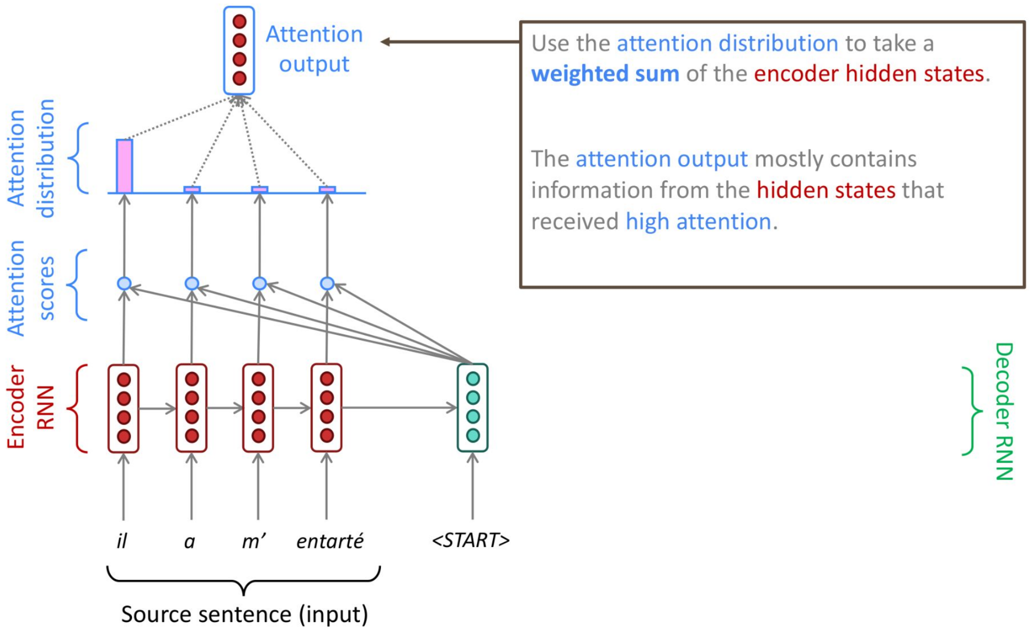
LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

Part 1

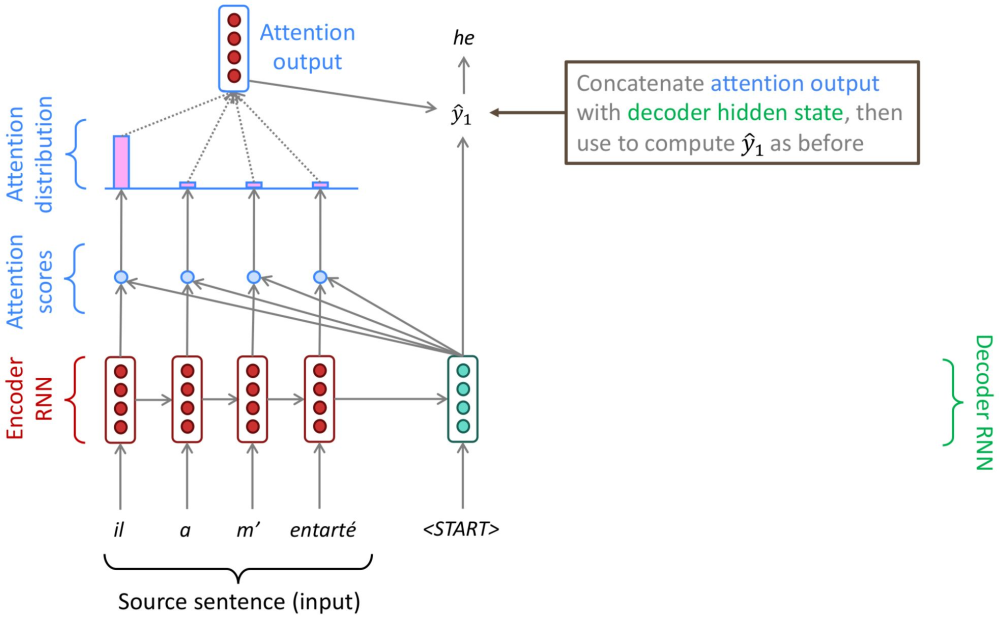
LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

Part 1

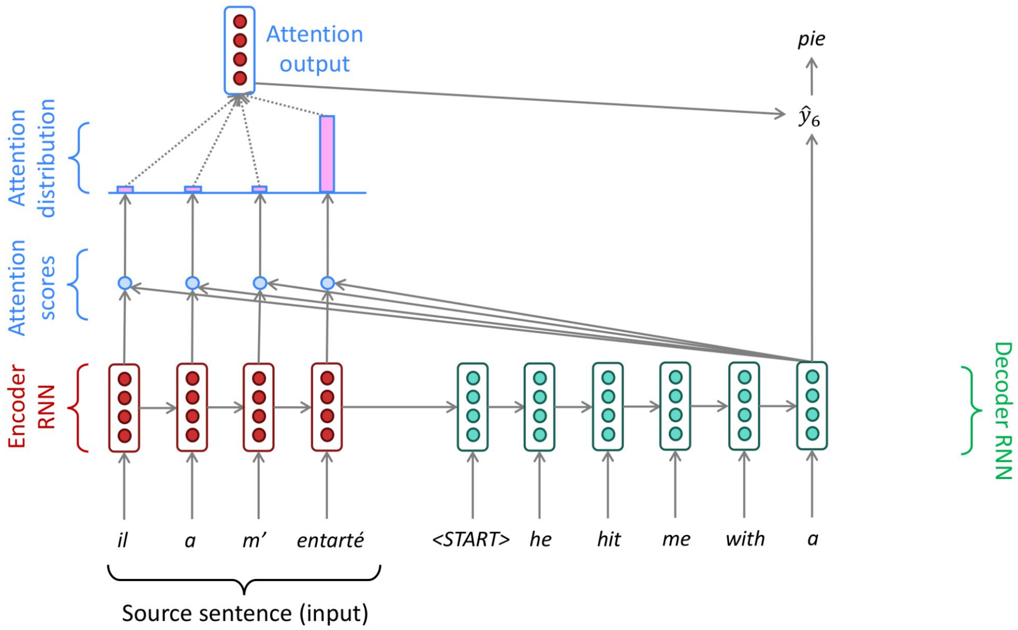
LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

Part 1

LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

Why Attention?

- Attention significantly improves Translation performance:
  It's very useful to allow decoder to focus on certain parts of the source
- Attention provides a more "human-like" model of the MT process:
  You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem:
  allows decoder to look directly at source
- Attention helps with the vanishing gradient problem:
  Provides shortcut to faraway states

GRU+Attention for Machine Translation

Source: An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.

Reference: Le privilège d'admission est le droit d'un médecin, en vertu de son statut de membre soignant d'un hôpital, d'admettre un patient dans un hôpital ou un centre médical afin d'y délivrer un diagnostic ou un traitement.

RNNenc-50: Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de sonétat de santé.

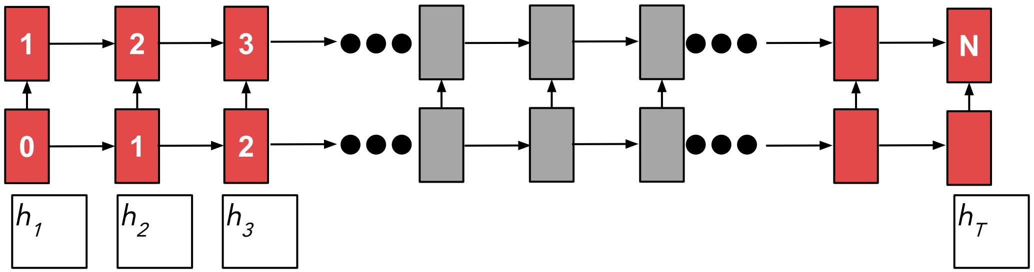
RNNsearch-50: Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.

Bahdanau et al. 2015

Part 1

RNN limit 3: Parallelization

- Forward and backward passes have O(sequence length) unparallelizable operations
- GPUs can perform many independent computations (like addition) at once!
- But future RNN hidden states can't be computed in full before past RNN hidden states have been computed.
- Training and inference are slow; inhibits on very large datasets!



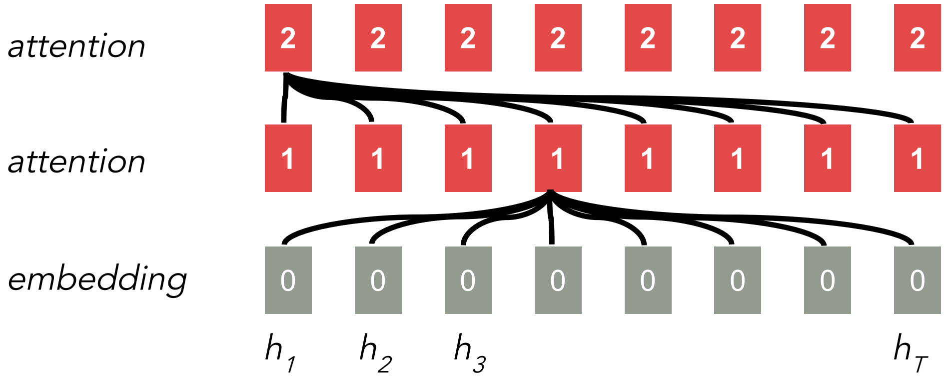
Numbers indicate min # of steps before a state can be computed

Break for questions and "appel"

Part 2

"Attention is all you need": Transformers

- Keeps only the attention mechanism, removes RNN
- Attention treats each token's representation as a query to access and incorporate information from a set of values.
- Number of unparallelizable operations does NOT increase with sequence length.

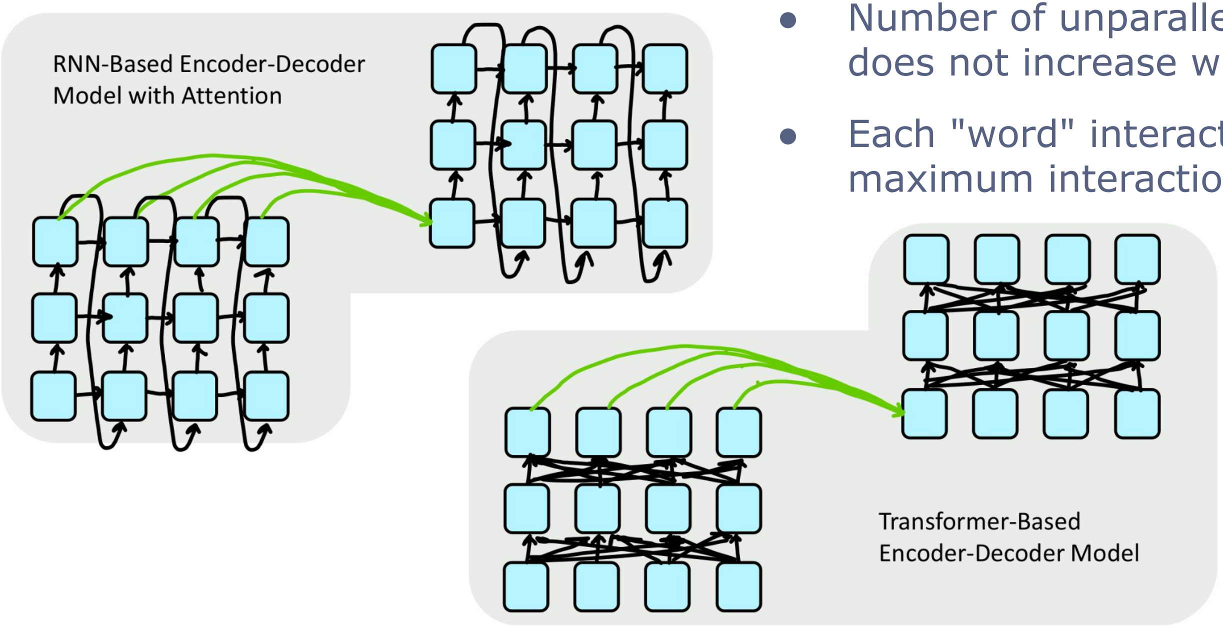


Maximum interaction distance: O(1), since all tokens interact at every layer!

All tokens attend to all tokens in previous layer; most arrows here are omitted

Recurrence vs. Attention

- Number of unparallelizable operations does not increase with sequence length.
- Each "word" interacts with each other, so maximum interaction distance is O(1).



Part 2

Deep Learning is made out of GPUs

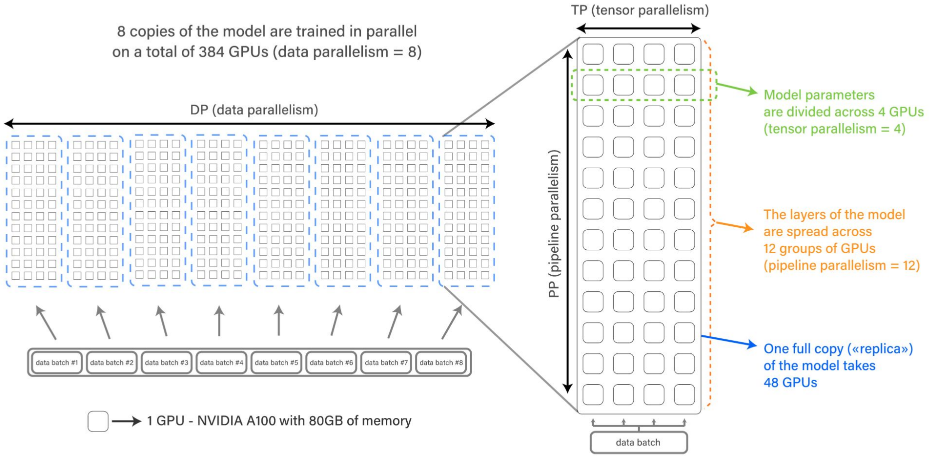
Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.

By Michael Kan January 18, 2024

(David Paul Morris/Bloomberg via Getty Images)

Deep Learning is made out of GPUs



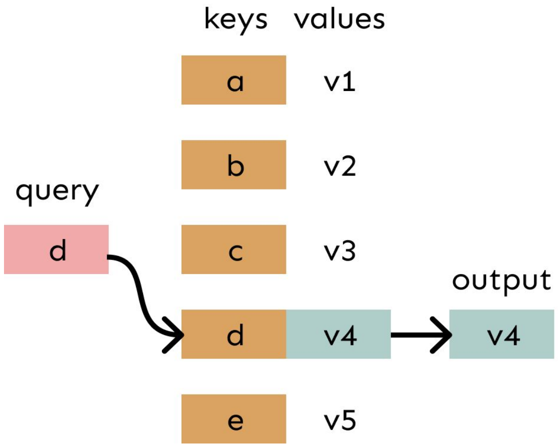
+38,400% in 12 years

Part 2

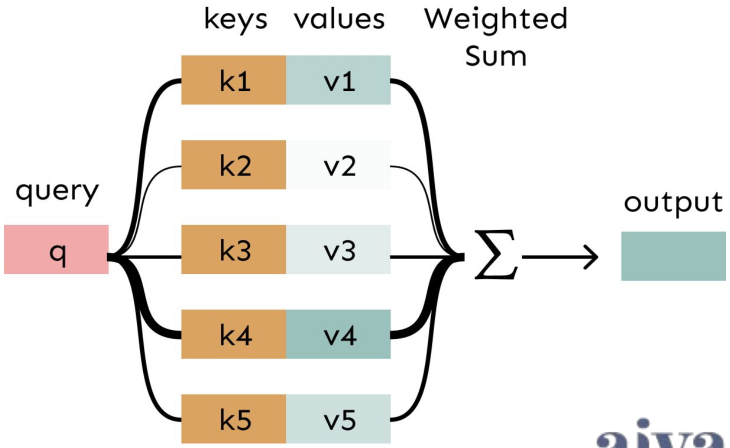
Attention as a soft, averaging lookup table

We can think of attention as performing fuzzy lookup in a key-value store.

In a **lookup table**, we have a table of keys that map to values. The query matches one of the keys, returning its value.

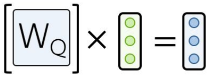


In **attention**, the query matches all keys *softly*, to a weight between 0 and 1. The keys' values are multiplied by the weights and summed.



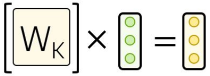
Self-Attention: Basic Concepts

Each vector receives three representations ("roles")



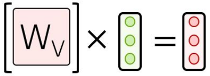
Query: vector **from** which the attention is looking

"Hey there, do you have this information?"



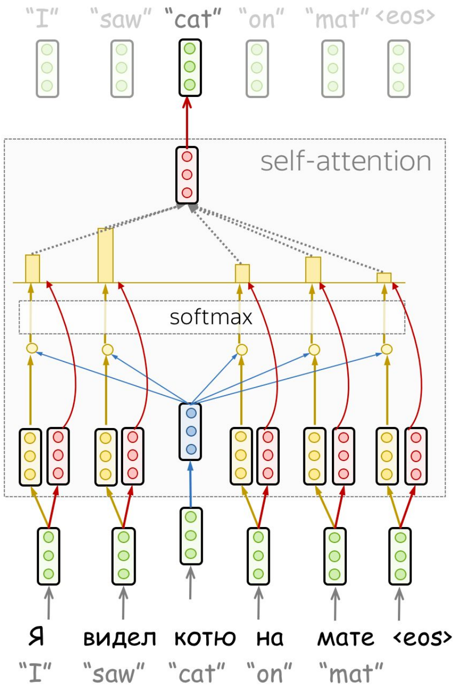
Key: vector **at** which the query looks to compute weights

"Hi, I have this information – give me a large weight!"



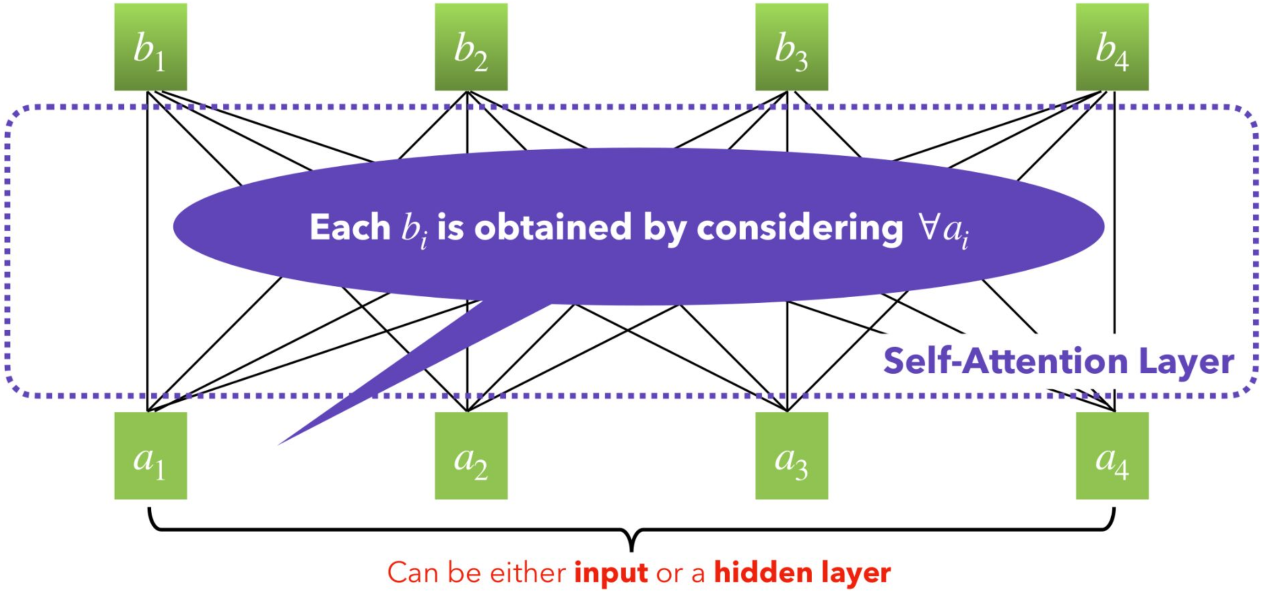
Value: their weighted sum is attention output

"Here's the information I have!"

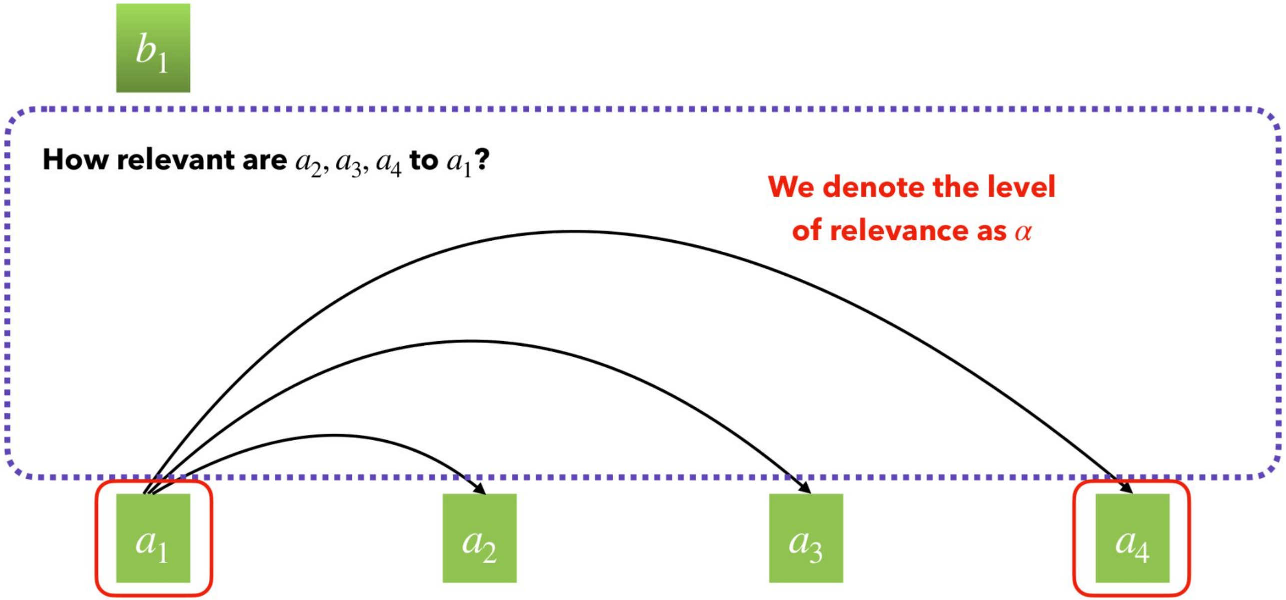


Part 2

Self-Attention: Walk-through

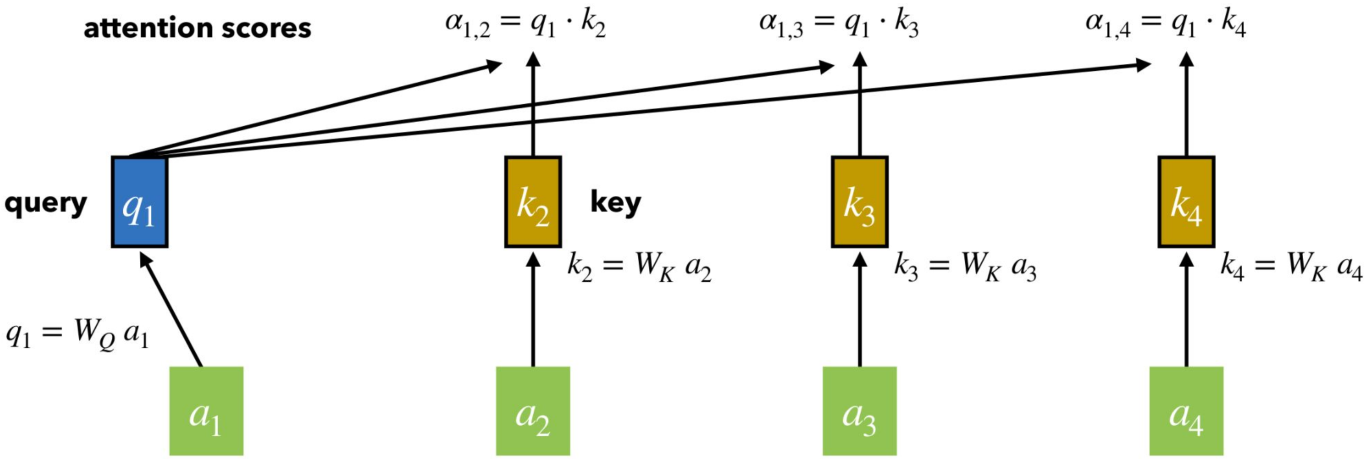


Self-Attention: Walk-through

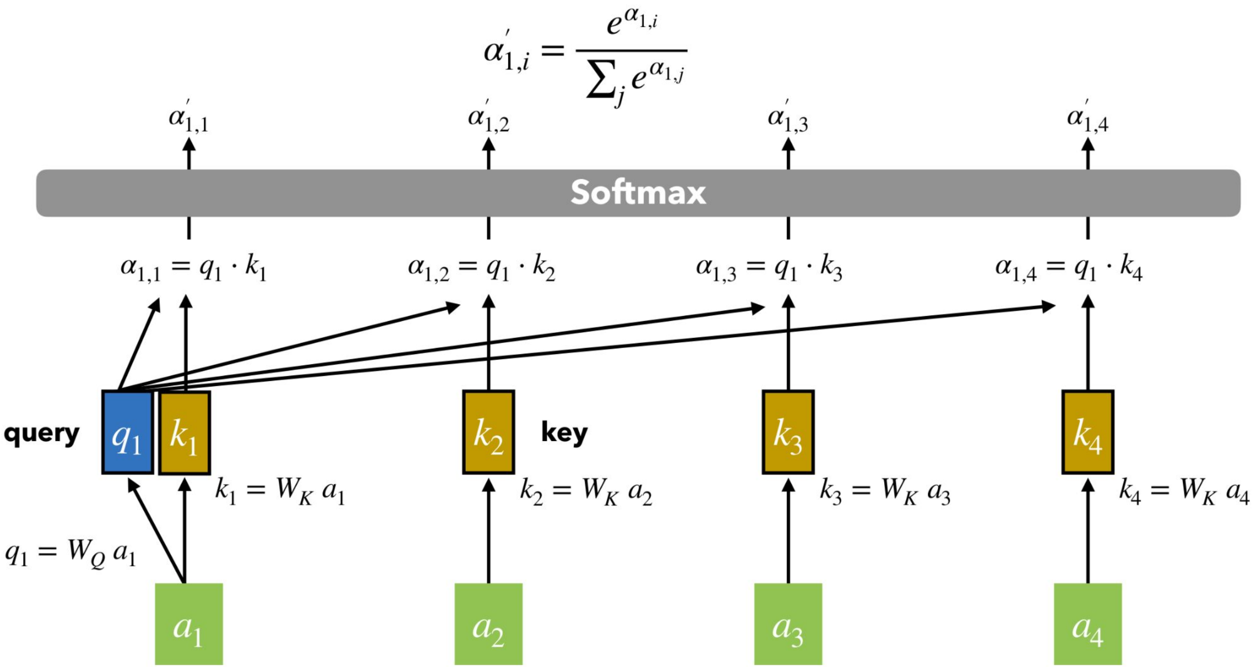


Part 2

Self-Attention: Walk-through

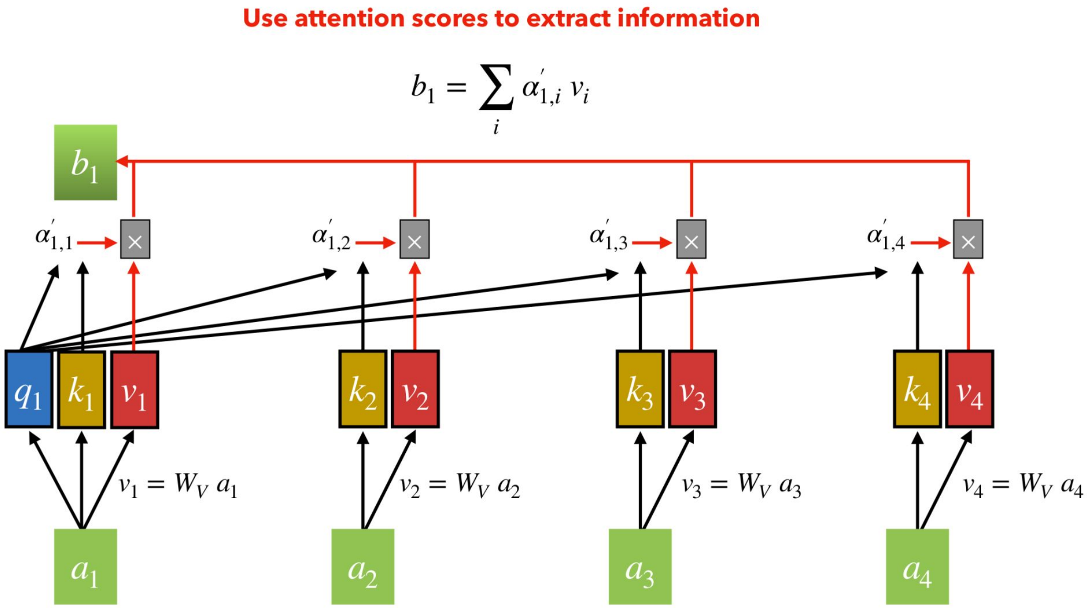


Self-Attention: Walk-through



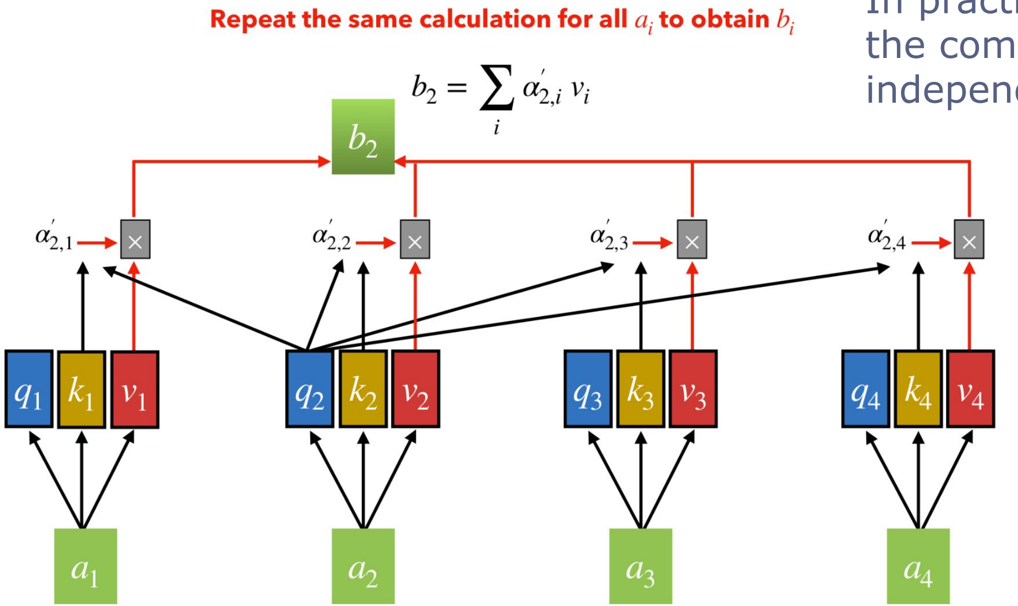
Part 2

Self-Attention: Walk-through



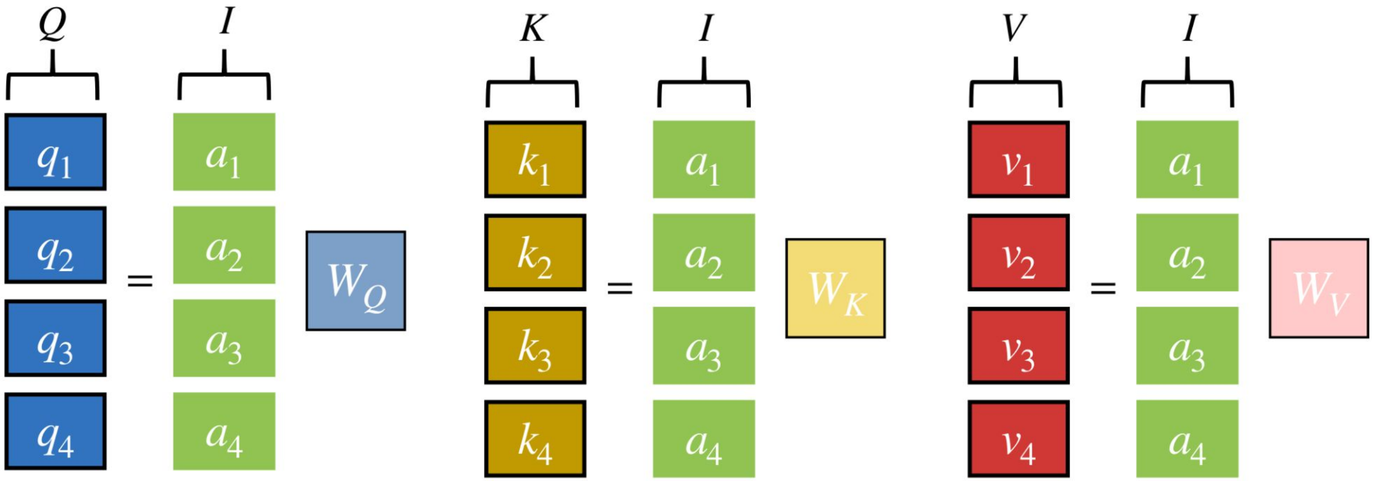
Part 2

Self-Attention: Walk-through



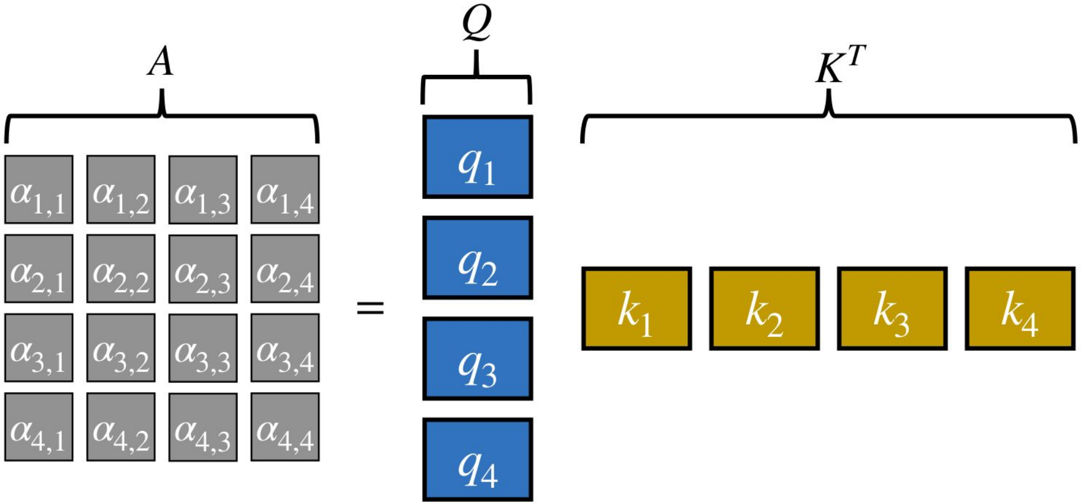
In practice this is done in **parallel**: the computation of ***b_1*** is independent of ***b_2***

Self-Attention: in parallel



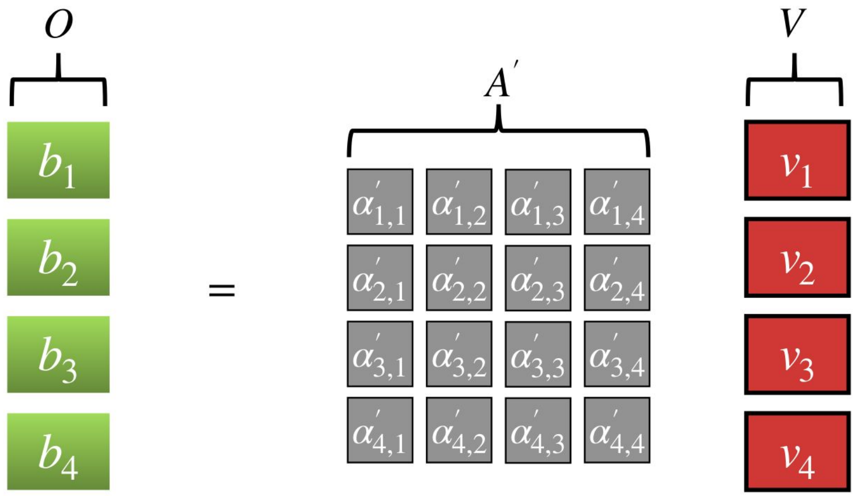
Part 2

Self-Attention: in parallel



Part 2

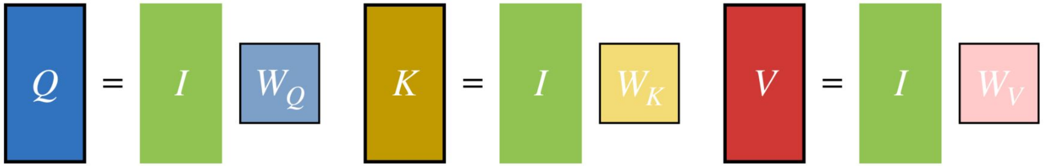
Self-Attention: in parallel



Part 2

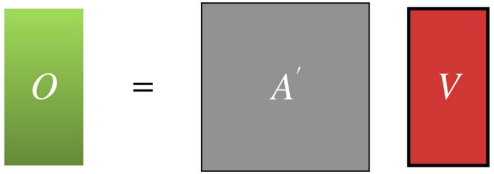
Self-Attention: in parallel

$$Q = I\, W_Q$$
$$K = I\, W_K$$
$$V = I\, W_V$$



$$A = Q\, K^T$$
$$A = I\, W_Q\, (I\, W_K)^T = I\, W_Q\, W_K^T\, I^T$$
$$A' = \text{softmax}(A)$$

$$O = A'\, V$$



Part 2

Self-Attention: formally

$$Q = I\,W_Q$$
$$K = I\,W_K$$
$$V = I\,W_V$$

$I = \{a_1, \ldots, a_n\} \in \mathbb{R}^{n \times d}$, where $a_i \in \mathbb{R}^d$

$W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$

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

$$A = Q\,K^T$$
$$A = I\,W_Q\,(I\,W_K)^T = I\,W_Q\,W_K^T\,I^T$$
$$A^{'} = \text{softmax}(A)$$

$A^{'}, A \in \mathbb{R}^{n \times n}$

$$O = A^{'}\,V$$

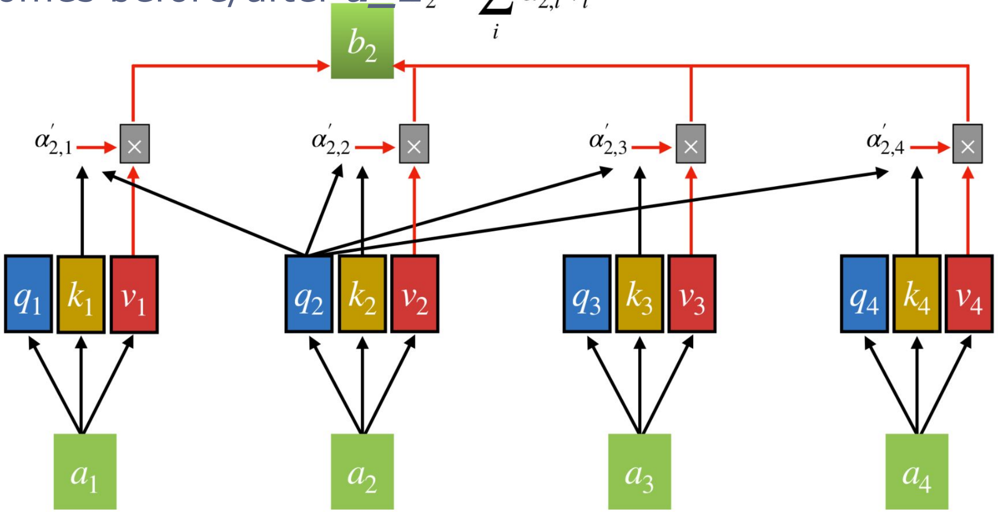
$O \in \mathbb{R}^{n \times d}$

Part 2

Permutation-invariant: Transformer = Bag of Word?

b_2 sums over all ***a_i***, does not matter if ***a_1*** comes before/after ***a_2***

$$b_2 = \sum_i \alpha'_{2,i} v_i$$



$\boldsymbol{x}_1$: `yes , we have no bananas`

$\boldsymbol{x}_2$: `say yes for bananas`

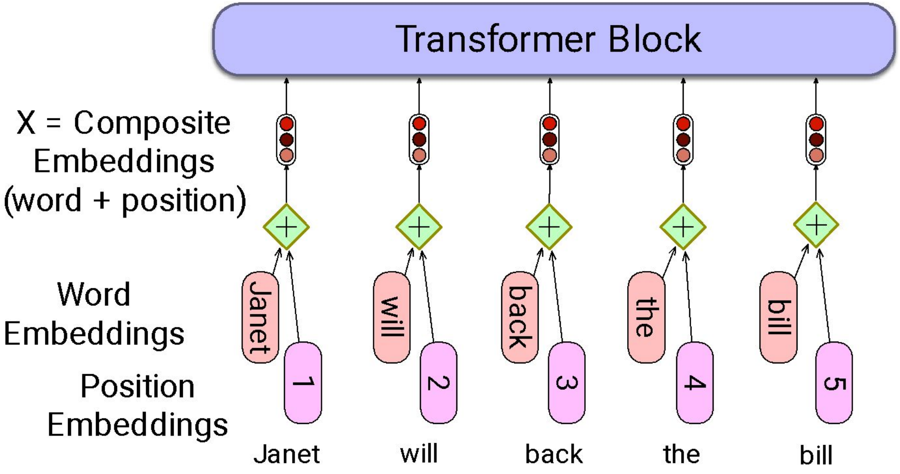
$\boldsymbol{x}_3$: `no bananas , we say`

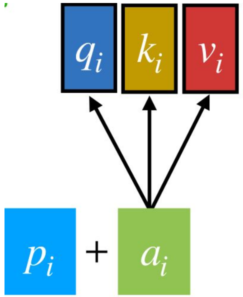
	1	2	3
,	1	0	1
bananas	1	1	1
for	0	1	0
have	1	0	0
no	1	0	1
say	0	1	1
we	1	0	1
yes	1	1	0

Part 2

Position Encoding

- Most basic method: At the 1st layer, add an embedding of the position to the word embedding (BERT, GPT-3)
- Typically initialized randomly and learned like any other parameter of the model
- Despite adding position, several papers argue that Transformer models are **permutation invariant** / do not model the order of words
- More recent methods we won't cover modify self-attention (RoPE, ALiBi)

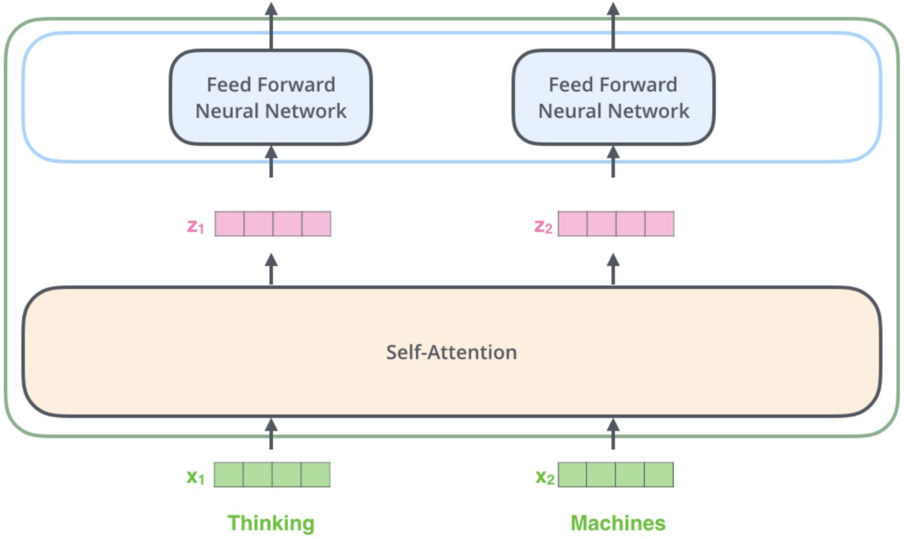




Part 2

Attention is almost all you need

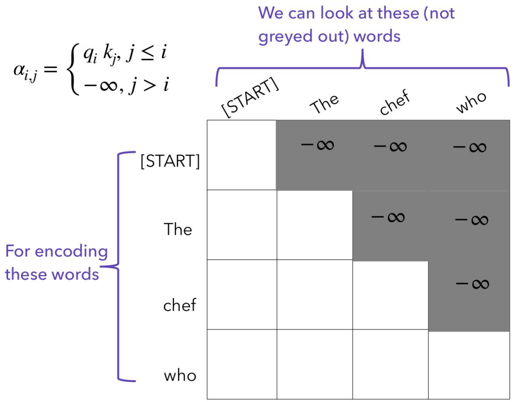
- Since there are no element-wise non-linearities, self-attention is simply performing a re-averaging of the value vectors
- Apply a feedforward layer to the output of attention, providing non-linear activation (and additional expressive power)

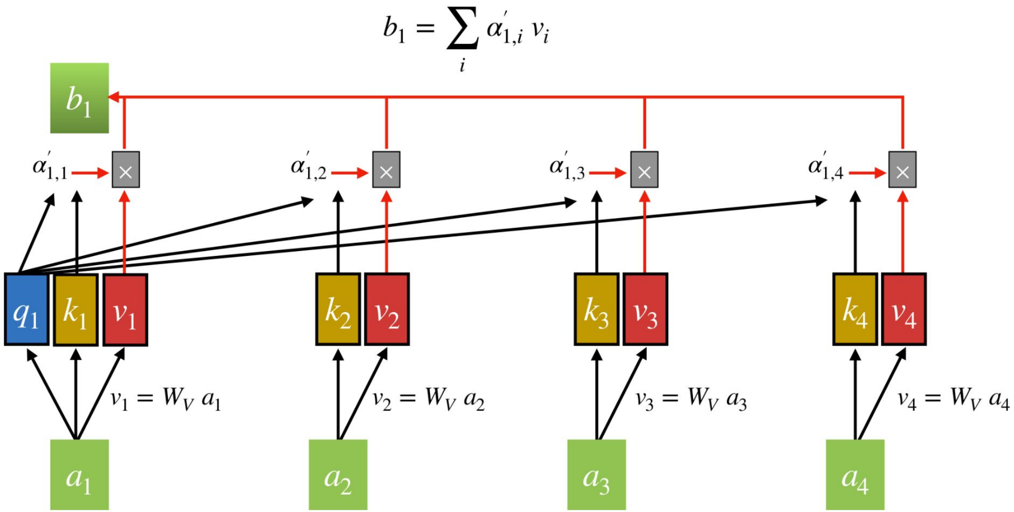


Part 2

Self-attention? What about causality?

- ***b_1*** depends on ***a_2*** and ***a_3***… but the goal is the **generate *a_2*** and ***a_3***
- **Mask** attention scores of future words to preserve **causality**

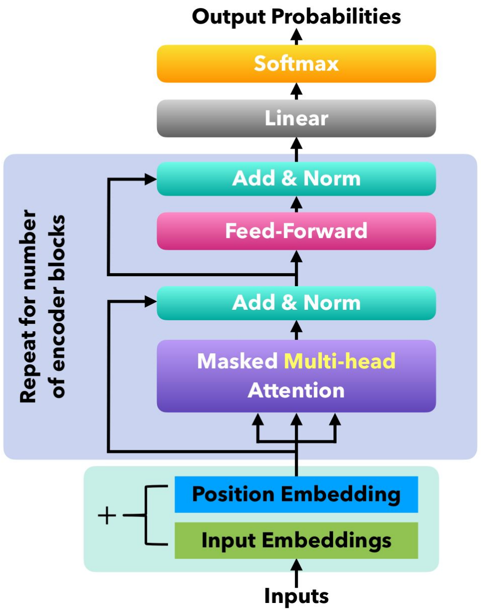




Part 2

Putting the pieces together

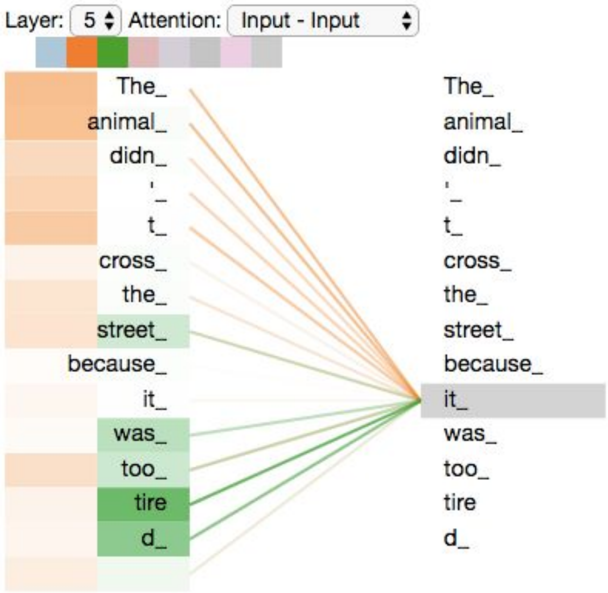
- Positional Encoding: otherwise permutation-invariant
- *(Multi-head)* Self-attention: essential part to model relations between words
    - masked for causality
- *Residual Connection:* for stable training/deeper networks
- Feedforward for nonlinearity/expressiveness
- Linear/softmax: output layer back to vocabulary dimension



Part 2

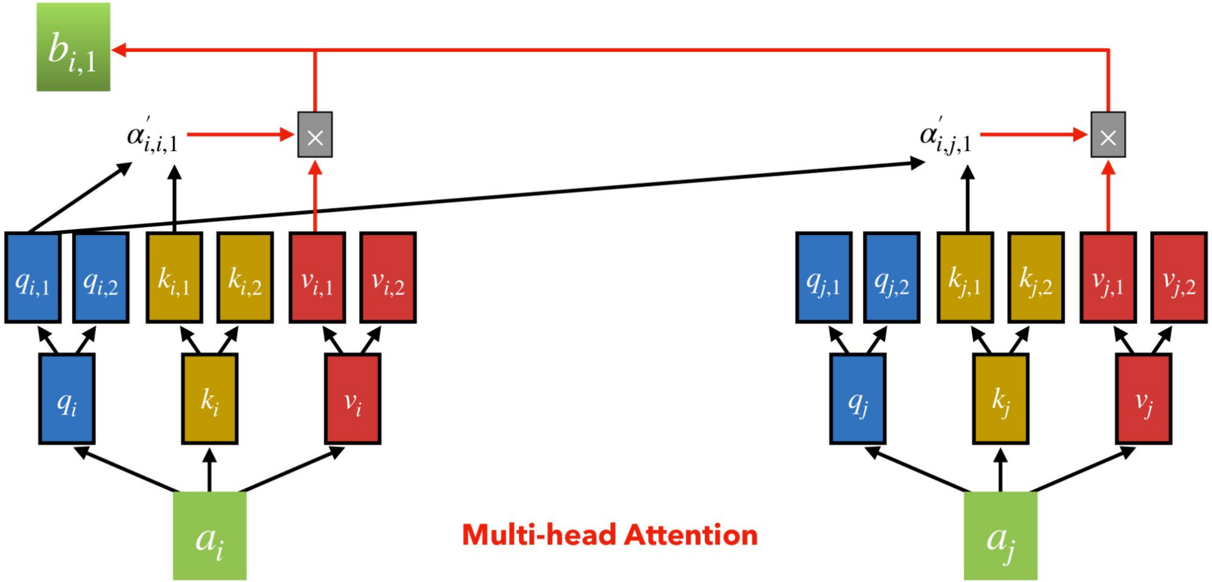
Why Multi-Head Attention?

- What if we want to look in multiple places in the sentence at once?
- For word i, self-attention "looks" where $x_i Q_i K_j$ is high, but maybe we want to focus on different j for different reasons?
- orange head for the coreference "The animal"
- green head for the past participle "tired"



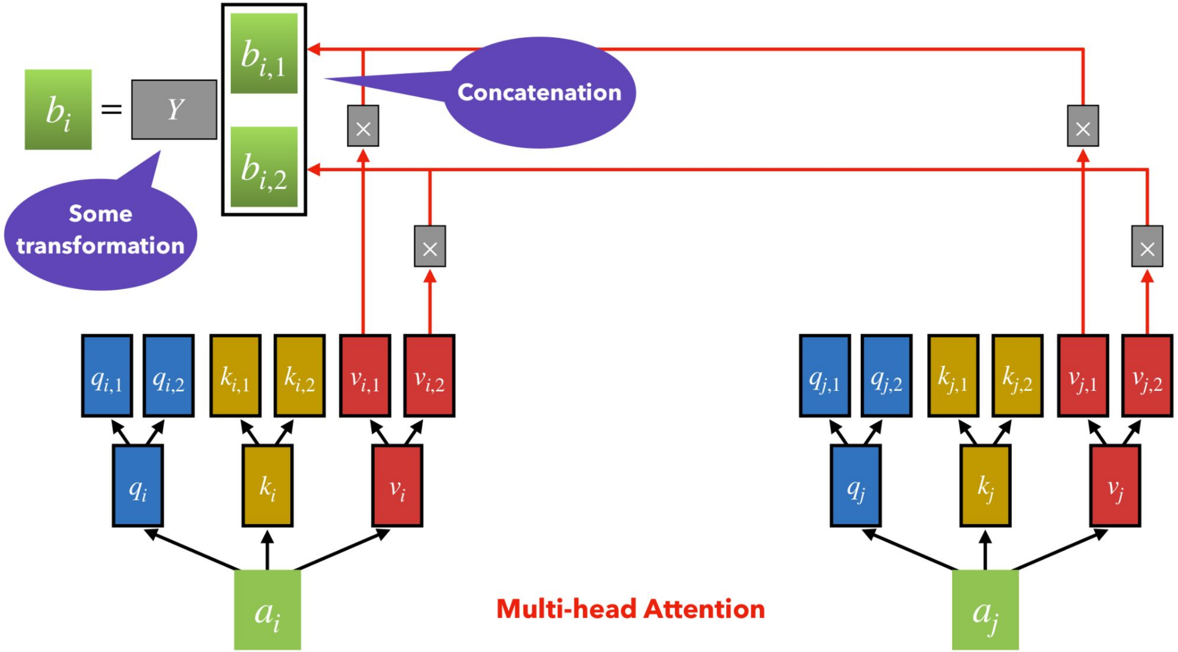
Part 2

Multi-Head Attention: Walk-through



Multi-head Attention

Multi-Head Attention: Walk-through



Part 2

Multi-Head Attention: formally

$Q^l = I\, W_Q^l$

$K^l = I\, W_K^l$

$V^l = I\, W_V^l$

$I = \{a_1, \ldots, a_n\} \in \mathbb{R}^{n \times d}$, where $a_i \in \mathbb{R}^d$

$W_Q^l, W_K^l, W_V^l \in \mathbb{R}^{d \times \frac{d}{h}}$

$Q^l, K^l, V^l \in \mathbb{R}^{n \times \frac{d}{h}}$

Multiple attention “heads” can be defined via multiple **W*** matrices

$A^l = Q^l\, K^{l^T}$

$A^{l'} = \text{softmax}(A^l)$

$A^{l'}, A^l \in \mathbb{R}^{n \times n}$

$O^l = A^{l'}\, V^l$

$O^l \in \mathbb{R}^{n \times \frac{d}{h}}$

Each attention head performs attention independently

$O = [O^1; \ldots; O^h]\, Y$

$Y \in \mathbb{R}^{d \times d}$

$[O^1; \ldots; O^h] \in \mathbb{R}^{n \times d}$

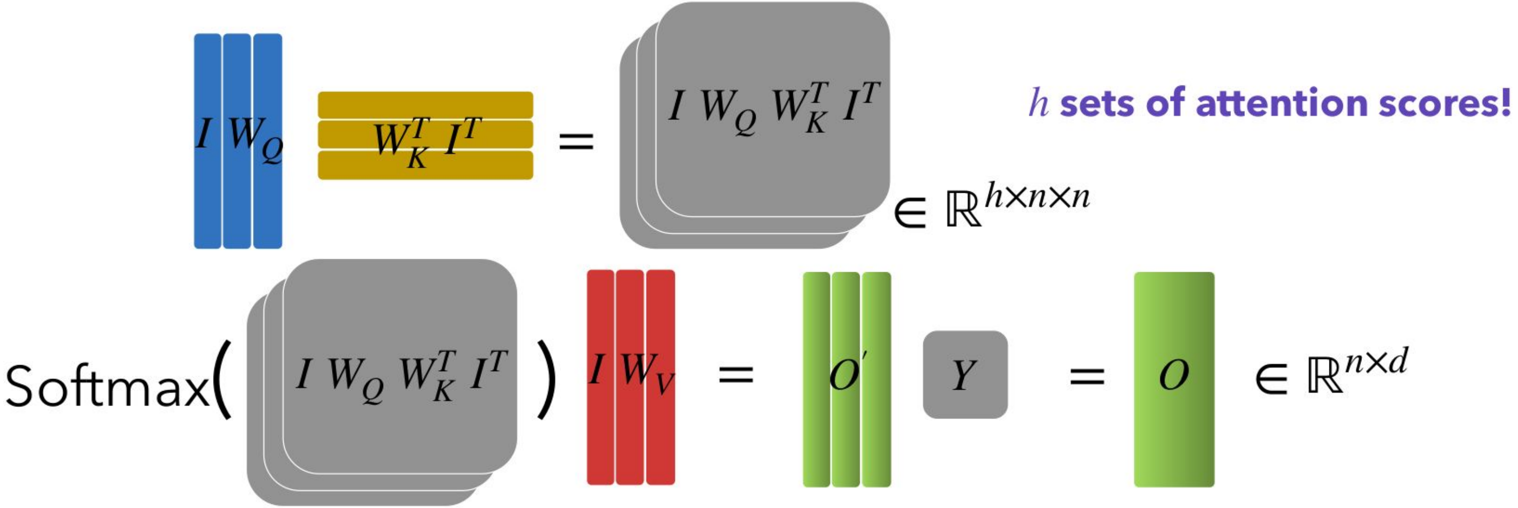
$O \in \mathbb{R}^{n \times d}$

Their results are concatenated

Multi-Head Attention: in parallel (as always)

compute $I\,W_Q \in \mathbb{R}^{n \times d}$, and then reshape to $\mathbb{R}^{n \times h \times \frac{d}{h}}$

Then we transpose to $\mathbb{R}^{h \times n \times \frac{d}{h}}$; **now the head axis is like a batch axis**



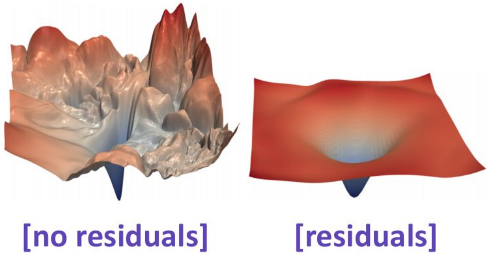
Residual Connections for stable training

- Residual connections are a trick to help models train better.
  - Instead of $X^{(i)} = \text{Layer}(X^{(i-1)})$ (where i represents the layer)

  - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$ (so we only have to learn "the residual" from the previous layer)

  - Gradient is great through the residual connection; it's 1!
  - Bias towards the identity function!

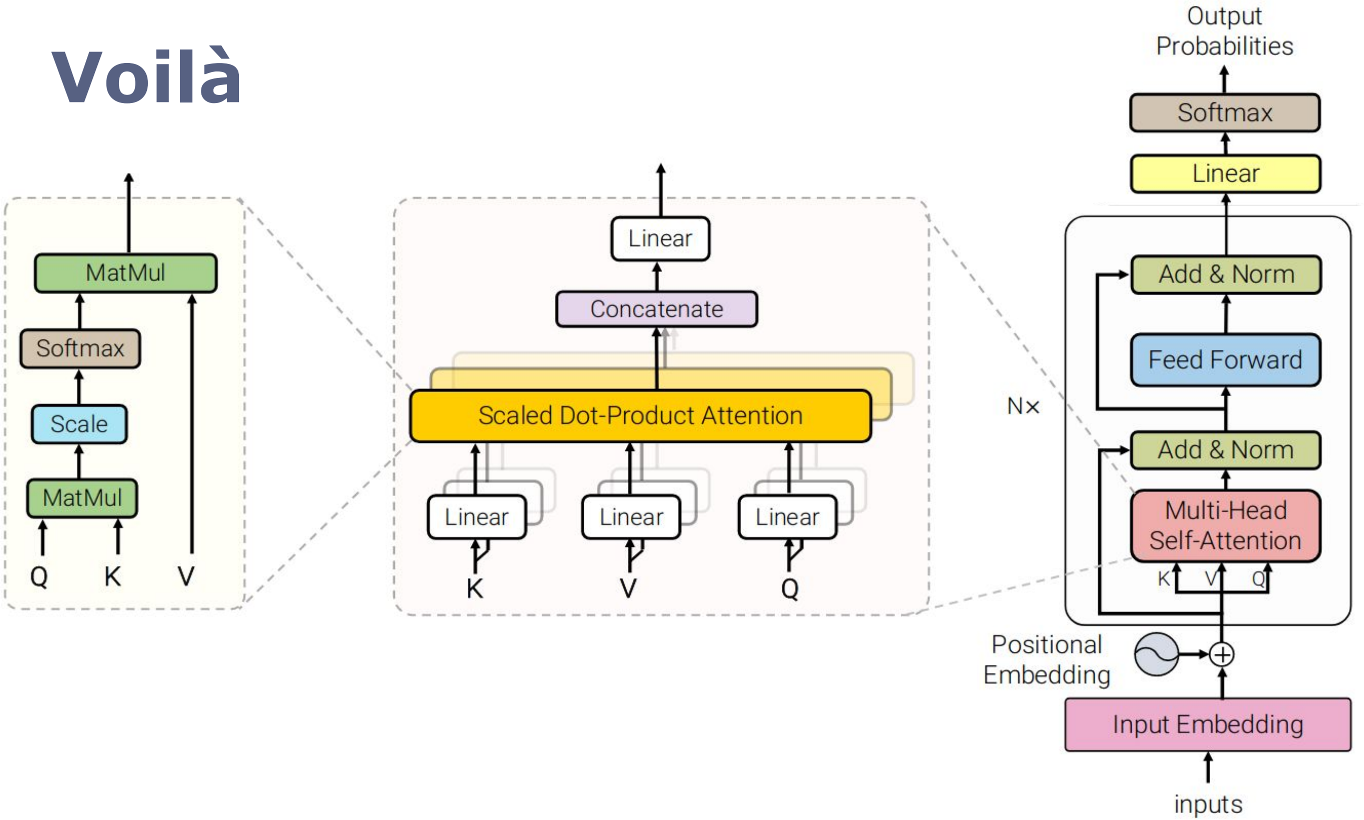
Remember the Cell state in LSTM



[Loss landscape visualization,
Li et al., 2018, on a ResNet]

Part 2

Voilà



MatMul
Softmax
Scale
MatMul
Q
K
V
Linear
Concatenate
Scaled Dot-Product Attention
Linear
Linear
Linear
K
V
Q
Output
Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head
Self-Attention
K
V
Q
Positional
Embedding
Input Embedding
inputs

Part 1

Transformer for Machine Translation

Source: An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.

Reference: Le privilège d’admission est le droit d’un médecin, en vertu de son statut de membre soignant d’un hôpital, d’admettre un patient dans un hôpital ou un centre médical afin d’y délivrer un diagnostic ou un traitement.

RNNsearch-50: Un privilège d’admission est le droit d’un médecin d’admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l’hôpital.

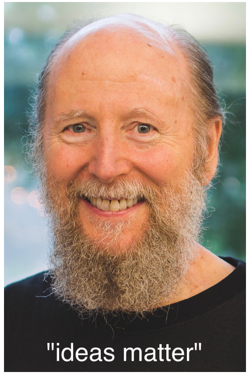
Transformer (fairseq wmt14.en-fr): Un privilège d'admission est le droit d'un médecin d'admettre un patient dans un hôpital ou un centre médical pour y effectuer un diagnostic ou une intervention, en fonction de son statut de travailleur de la santé dans un hôpital.

Vaswani et al. 2017

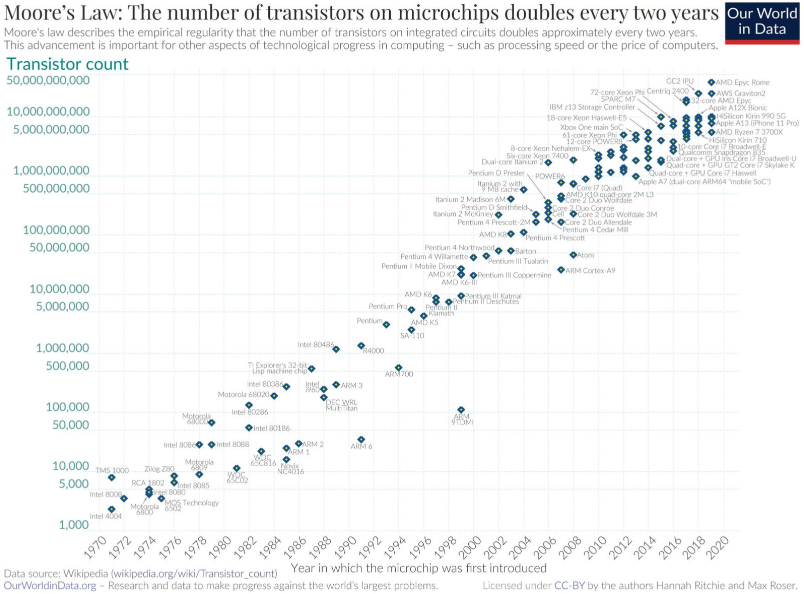
Part 2

Scaling Transformers (the bitter lesson)

"The biggest lesson that can be read from 70 years of AI research is that **general methods that leverage computation are ultimately the most effective**, and by a large margin"

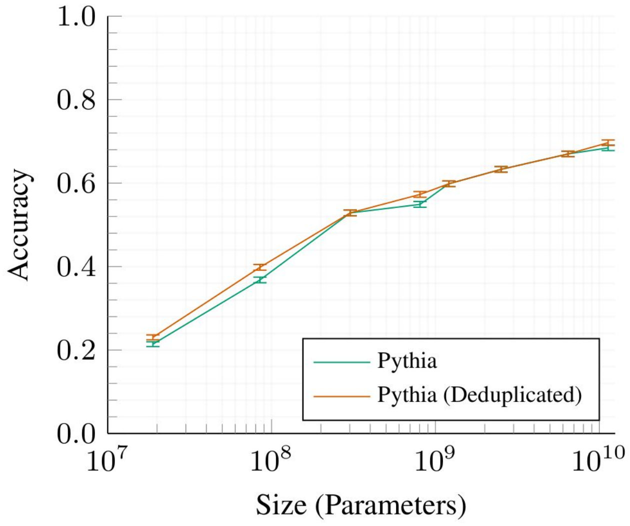


Rich Sutton



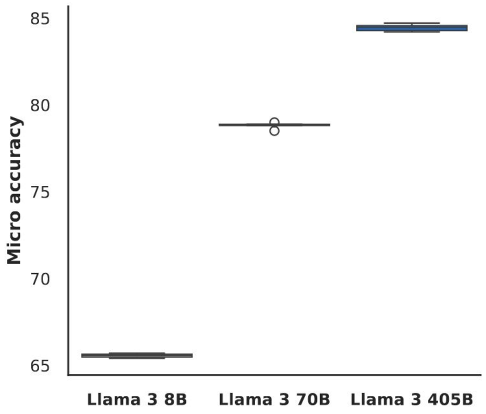
Part 2

Scaling Transformers (the bitter lesson)

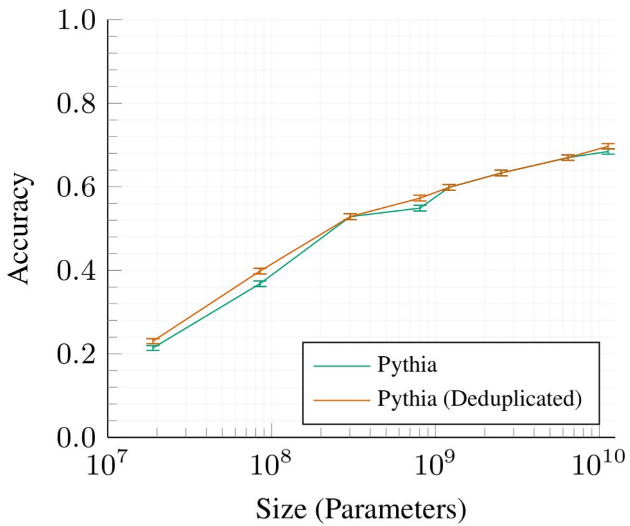


(a) LAMBADA (OpenAI)

There seems to be no limit, from millions to billions to trillions of parameters

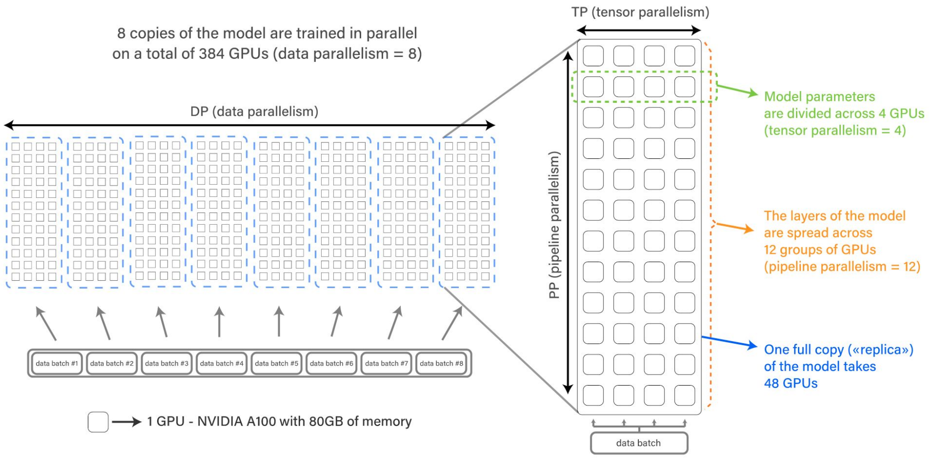


Scaling Transformers: buy more GPUs



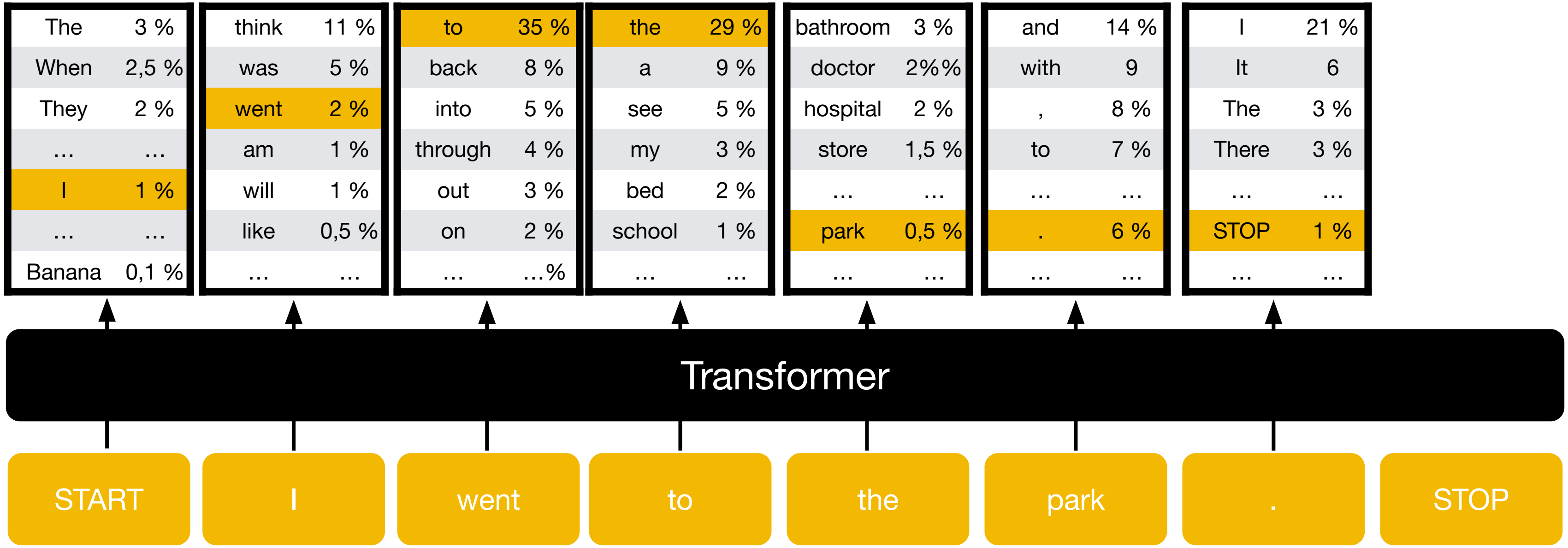
(a) LAMBADA (OpenAI)

From 8 × 32GB V100 GPUs (RoBERTa) to 16,384 × 94GB H100 GPUs (Llama 3)



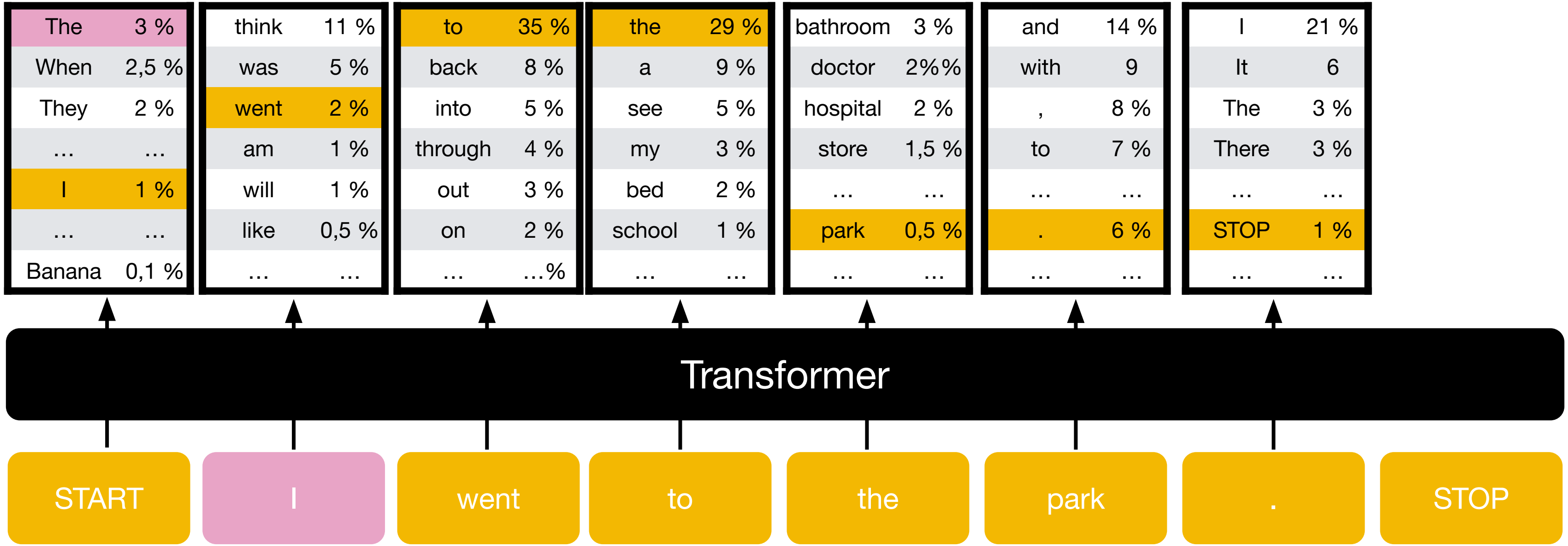
Part 2

Limit: Teacher Forcing



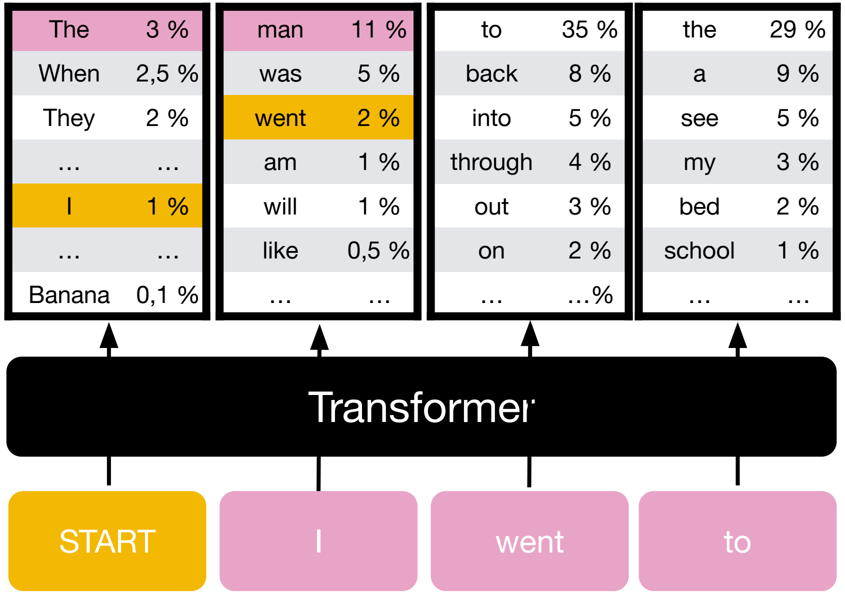
Part 2

Train-test mismatch: exposure bias



Part 1

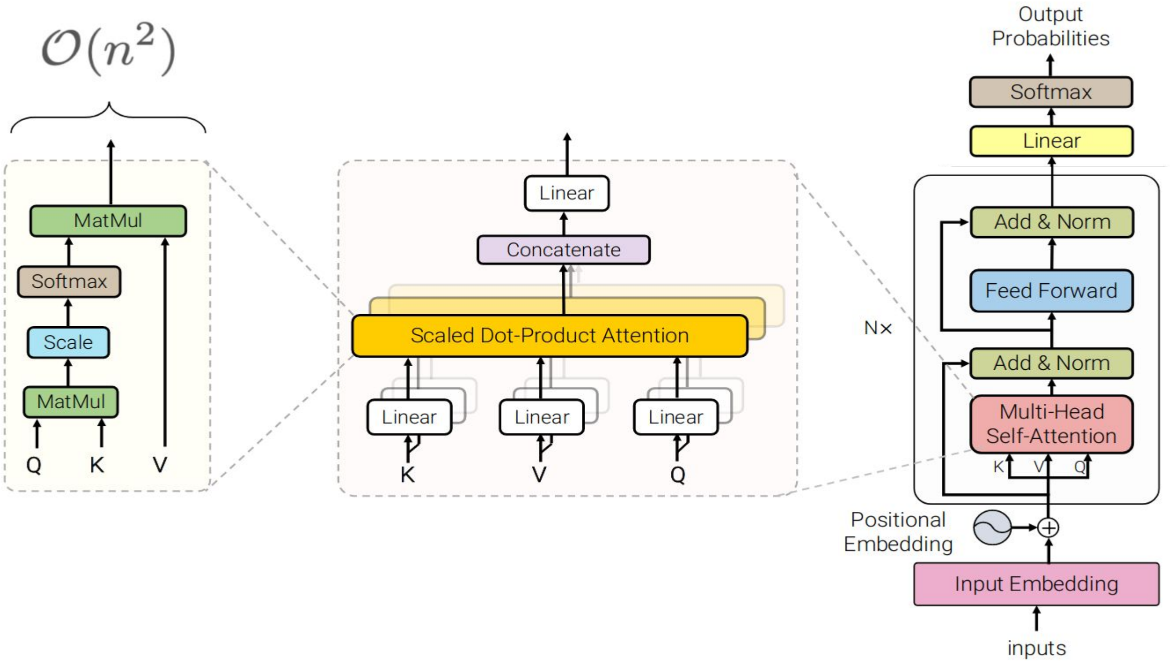
Not easily solved, unlike for RNN



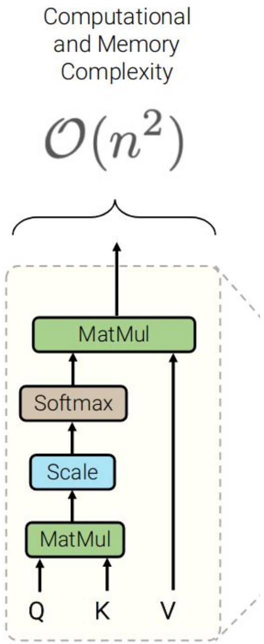
- Because computation is done in parallel, we cannot access the generation of the model
- Teacher forcing is done systematically, model are subject to exposure bias

Mihaylova, T., & Martins, A. F. T. (2019). Scheduled Sampling for Transformers.

Limit: Quadratic complexity



Limit: Quadratic complexity

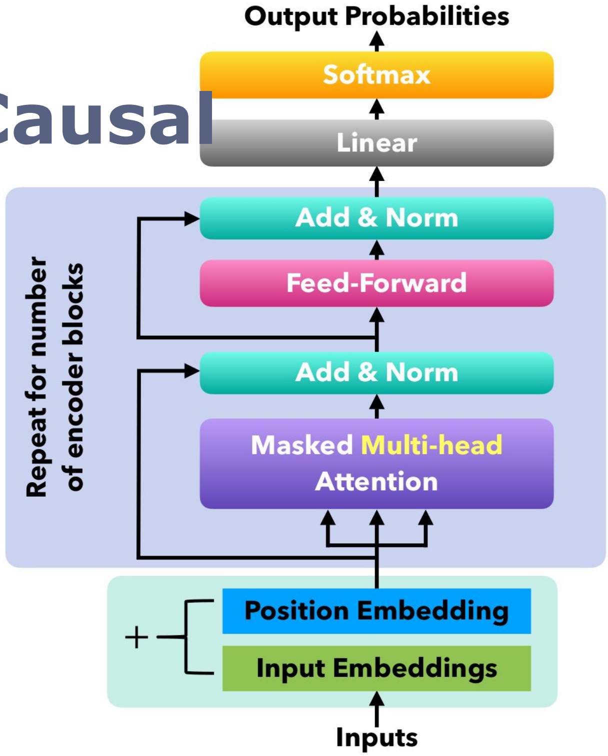


- Computing all pairs of interactions means our computation grows **quadratically** with the sequence length!
- for recurrent models, it only grew **linearly**
- Large body of work on this question (Tay et al., 2020) "Efficient Transformers: A Survey"
- But vanilla Transformer still used in state-of-the-art LLMs

Part 2

Transformer A: (Autoregressive) Decoder/Causal

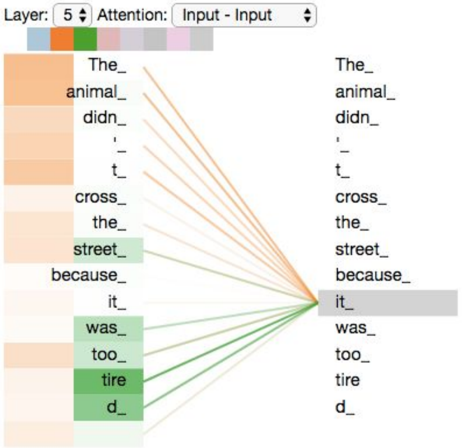
- Described previously: main architecture for LLMs (**GPT-3**, Llama-*, and many many many more)
- **Causal**/unidirectional mask: can only see past words
- First purpose: **Language Modeling** / autoregressive generation
- But now every task of NLP is cast as Language Modeling, even classification

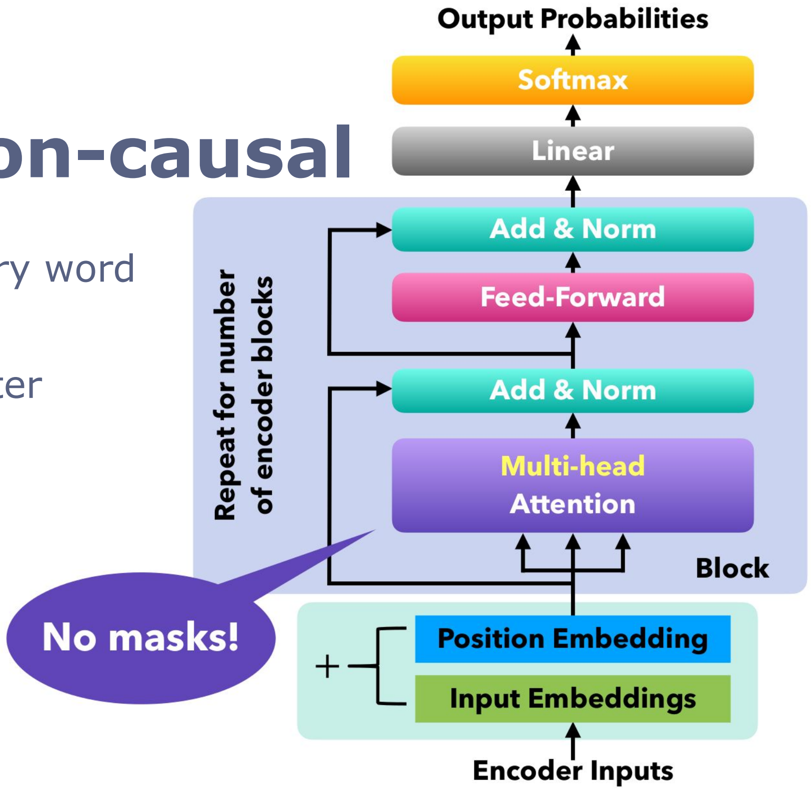


Part 2

Transformer B: (Bidirectional) Encoder/non-causal

- Removes the mask from self-attention: now every word can see future and past
- Use for classification (but now words have a better context, unlike bag of words)
- Famous examples: **BERT**, m**BERT**, Ro**BERT**a, De**BERT**a, Camem**BERT**, …

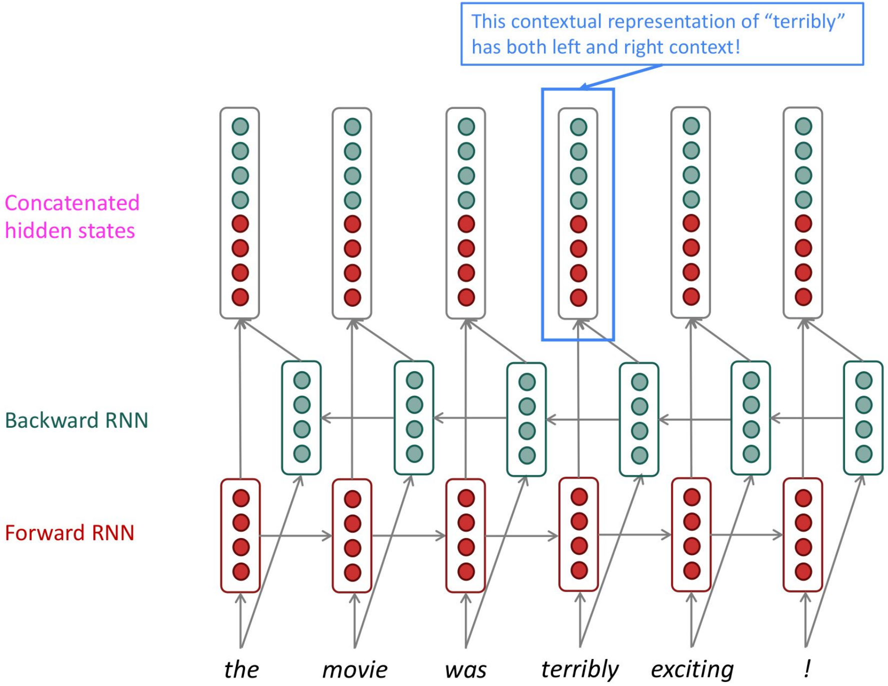




Part 2

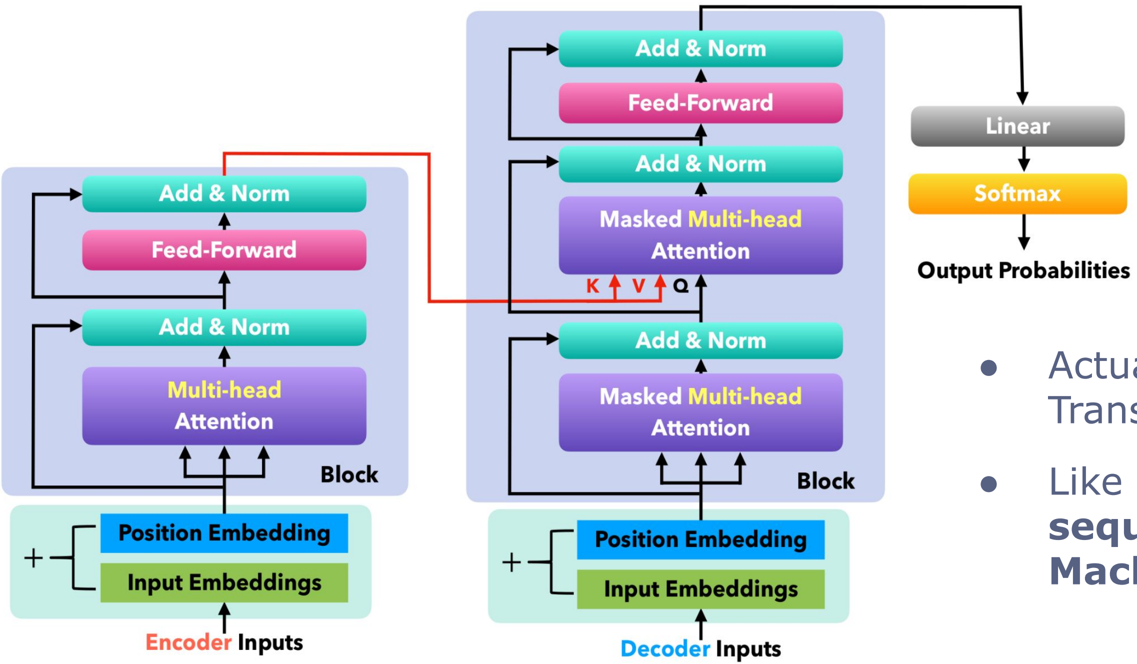
Note on Bidirectional RNNs

- RNNs could also be bidirectional but you then needed two!
- → long sequential (unparallelizable) operations, although still ***O(n)*** theoretically



Part 2

Transformer C: Encoder-Decoder

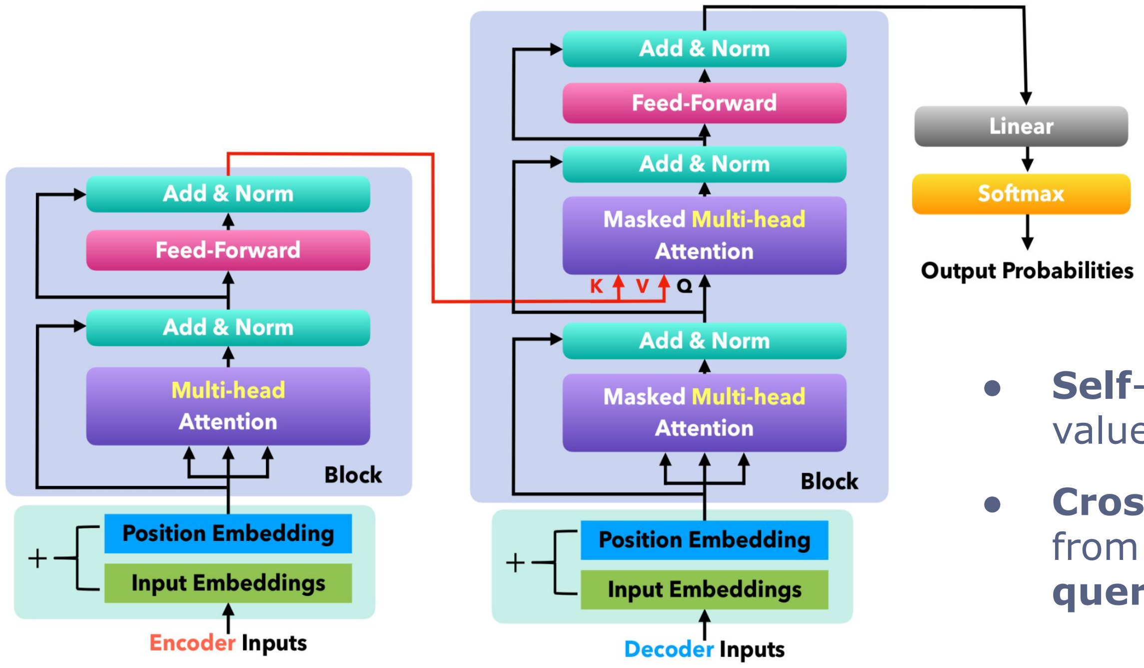


- Famous examples: T5, BART, BARThez, …

- Actually the first variant proposed for Translation by Vaswani et al. (2017)
- Like an RNN Encoder-Decoder, use for **sequence-to-sequence** tasks like **Machine Translation**

Part 2

Cross-Attention in Encoder-Decoder



- **Self**-attention: queries, keys, and values come **from the same source**
- **Cross**-Attention: *keys* and *values* are from *Encoder* (like a memory); **queries** are from **Decoder**

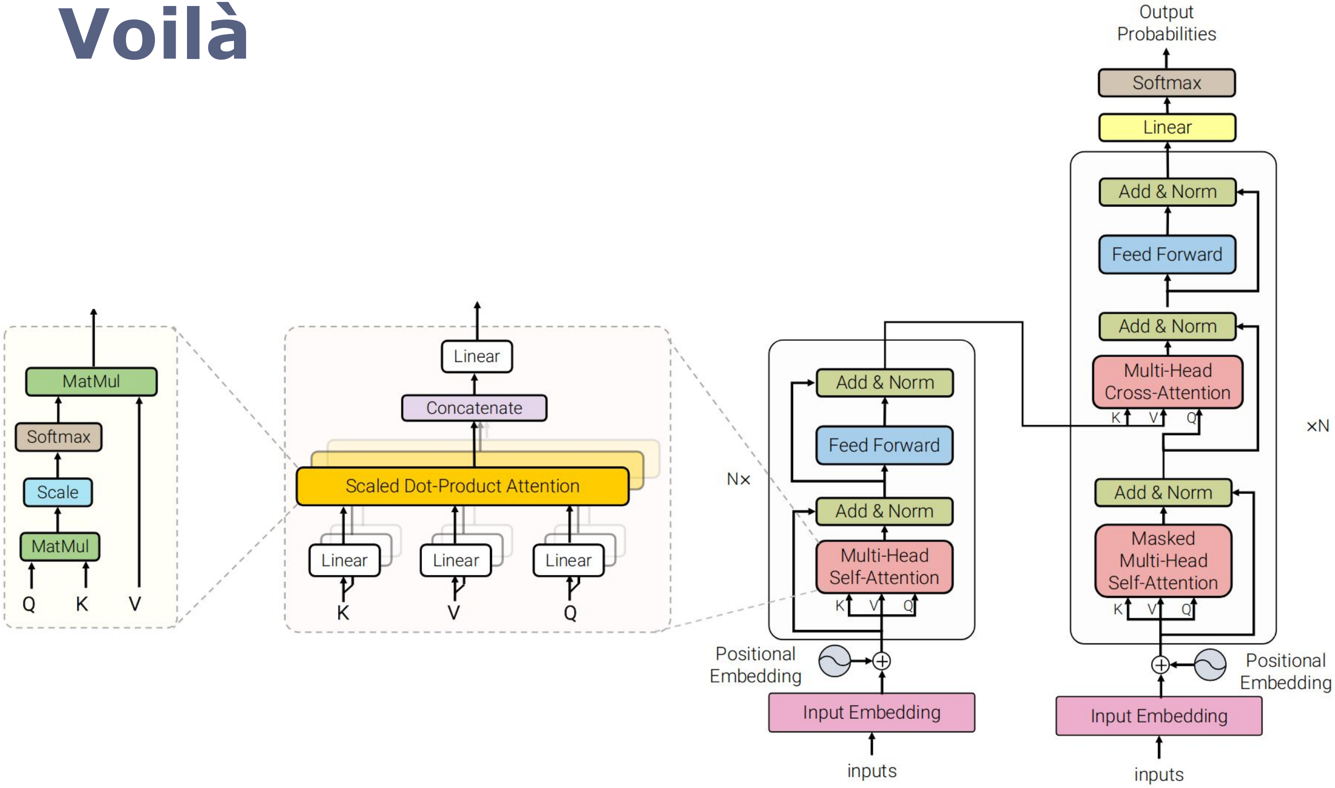
Part 2

Summarizing

- Use Transformers for:
  - Sequence-to-sequence (e.g. Translation)
  - Natural Language Generation/Language Modeling
  - Actually anything: encode text (with context) then classify
- Attention was invented for RNNs and Translation, to focus on parts of the source sentence: relieves information bottleneck
- Transformer leverages attention in a parallelized way: scales as long as you buy enough GPUs

Part 2

Voilà



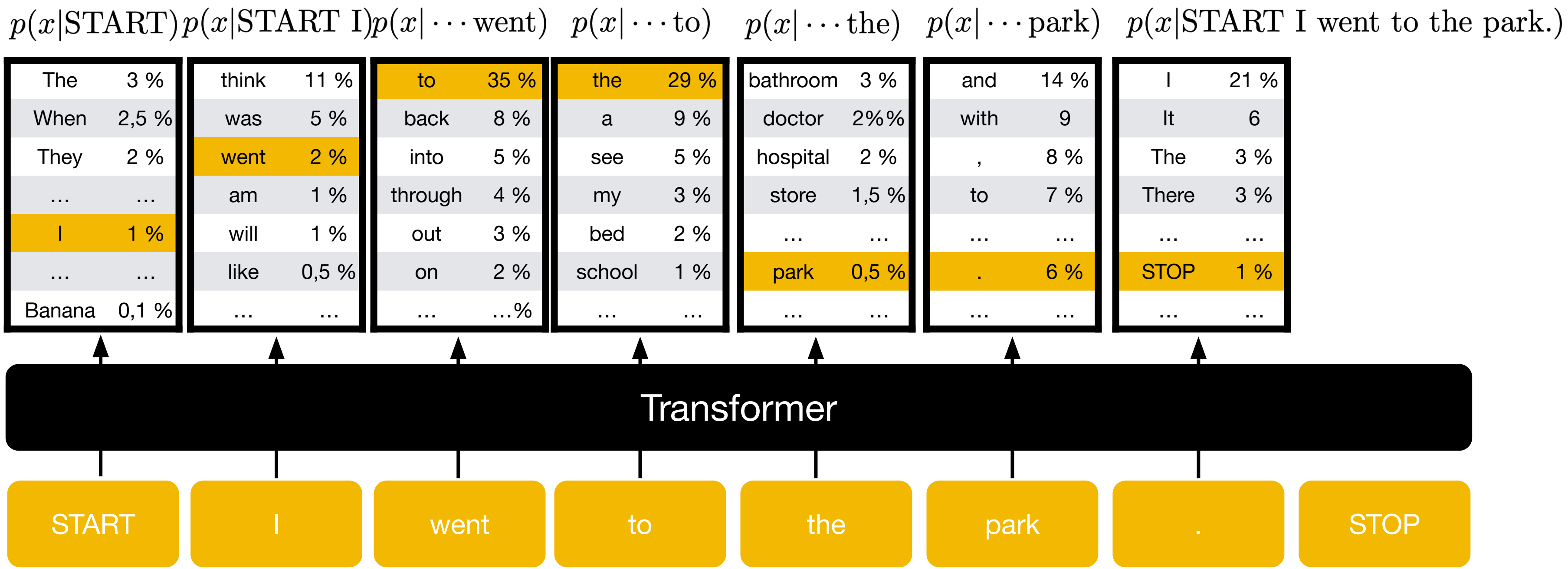
MatMul
Softmax
Scale
MatMul
Q
K
V
Linear
Concatenate
Scaled Dot-Product Attention
Linear
Linear
Linear
K
V
Q
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Cross-Attention
K V Q
×N
Add & Norm
Masked Multi-Head Self-Attention
K V Q
Positional Embedding
Input Embedding
inputs
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Self-Attention
K V Q
Positional Embedding
Input Embedding
inputs

Part 2

Homework!

Part 2

Next class: *(Pre-)Training* Transformers (LLMs)



Acknowledgements

This class directly builds upon:

- **Jurafsky, D., & Martin, J. H.** (2024). Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rdéd.).
- **Eisenstein, J.** (2019). Natural Language Processing. 587.
- **Yejin Choi**. (Winter 2024). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Noah Smith**. (Winter 2023). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Benoît Sagot**. (2023-2024). *Apprendre les langues aux machines* (Collège de France)
- **Chris Manning**. (Spring 2024). Stanford CS224N: Natural Language Processing with Deep Learning
- Classes where I was/am Teacher Assistant:
  - **Christopher Kermorvant**. Machine Learning for Natural Language Processing (ENSAE)
  - **François Landes** and **Kim Gerdes**. Introduction to Machine Learning and NLP (Paris-Saclay)

Also inspired by:

- My PhD thesis: *Répondre aux questions visuelles à propos d'entités nommées* (2023)
- **Noah Smith** (2023): Introduction to Sequence Models (LxMLS)
- **Kyunghyun Cho**: Transformers and Large Pretrained Models (LxMLS 2023), Neural Machine Translation (ALPS 2021)
- My former PhD advisors **Olivier Ferret** and **Camille Guinaudeau** and postdoc advisor **François Yvon**
- My former colleagues at LISN

aivancity

PARIS-CACHAN

advancing education
in artificial intelligence